J. BOISGONTIER • S. BREBION • G. FOUCAULT

Il COMMODORE 64
per tutti

Iniziazione + programmi

DELLO STESSO EDITORE

Volumi pubblicati

D. A. Lien - Dizionario del Basic
J. Boisgontier - Il Basic per tutti
A. Pinaud - CP/M passo dopo passo
D.-J. David - La scoperta del Commodore 64
D.-J. David - La pratica del Commodore 64
J. Deconchat - 102 programmi per Commodore 64
J. Boisgontier - Commodore 64: metodi pratici
X. Linant de Bellefonds - La pratica dello ZX Spectrum - Vol. 1
M. Henrot - La pratica dello ZX Spectrum - Vol. 2
J.-F. Séhan - Chiavi per lo ZX Spectrum
J. Lévy - Esercizi per lo ZX Spectrum
J.-F. Séhan - Alla ribalta: lo ZX Spectrum
C. Galais - Vademecum per Applesoft
J. Boisgontier - L'Apple e i suoi files
B. De Merly - Guida per l'Apple - Vol. 1
B. De Merly - Guida per l'Apple - Vol. 2
B. De Merly - Guida per l'Apple - Vol. 3
F. Lévy - Esercizi per l'Apple II, II plus, IIe, IIc
J. Boisgontier - 36 programmi per Apple IIe, II plus, IIc

Volumi di prossima pubblicazione

J. Deconchat - 102 programmi per ZX Spectrum e ZX 81
A. Pinaud - Programmare in Forth
N. Bréaud-Pouliquen - La pratica dell'Apple II - 1. Periferiche e gestione dei file
D.-J. David - La pratica del Commodore 64 - Linguaggio macchina e assembler del 6502
J.-P. Blanger - Modelli d'espressione grafica
J. Deconchat, V. Grandis - 102 programmi per il Philips C7420 Videopac +
C. Bardon, B. De Merly - Giochi su Philips C7420 Videopac +

JACQUES BOISGONTIER - SOPHIE BRÉBION
GÉRARD FOUCAULT

Il COMMODORE 64 per tutti

Iniziazione + programmi

Edizione italiana a cura di
FRANCO POTENZA

Traduzione di
CRISTINA OMINI

Editsi - Editoriale per le scienze informatiche - S.r.l.
MILANO 1985

Sommario

Come si presenta il vostro computer?

Avete deciso di avvicinarvi al linguaggio BASIC, e per questo tirocinio avete scelto un Commodore 64.

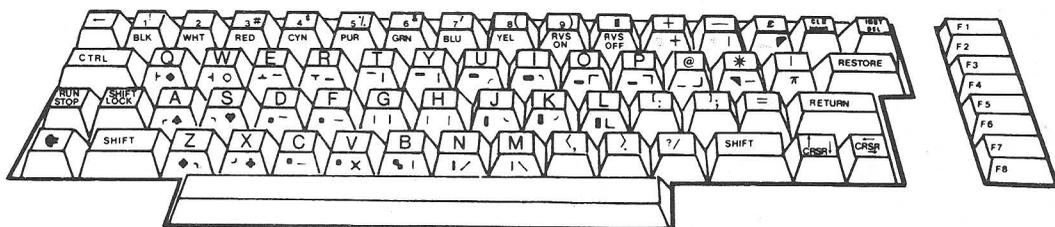
Lo schermo

Si tratta di uno schermo ben familiare, quello del vostro televisore, collegato al Commodore 64 attraverso la presa per apparecchiature esterne.
Una volta collegato, sullo schermo appare il seguente messaggio:

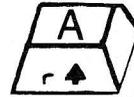
```
**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM  38911 BASIC BYTES FREE
```

Il computer dispone di 64 K di memoria, di cui 38911 bytes utilizzabili per il BASIC (64 K = 64.000 bytes).

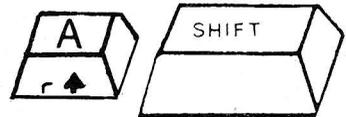
È attraverso lo schermo che il computer risponde alle vostre domande.
Ma come porre le domande al computer?
Non essendo il computer ancora in grado di accettare degli ordini vocali, dovete utilizzare un altro intermediario: la tastiera.

La tastiera

Come potete constatare, essa somiglia molto a quella di una macchina per scrivere: lettere, cifre, segni di interpunzione, simboli matematici, barra spaziatrice, c'è tutto.

A

Premendo il tasto "**A**" da solo, appare il carattere "A".

SHIFT

Premendo SHIFT, si accede ai caratteri di destra dei tasti.

Esempio: se premete simultaneamente "SHIFT" e "A", appare il carattere "↑".

Premendo " C= " si accede ai caratteri di sinistra dei tasti.

Esempio: nell'esempio qui a fianco, appare il carattere "┌".

Questo codice di caratteri si chiama "**MAIUSCOLO/GRAFICO**".

SHIFT + **C=**

Premendo simultaneamente SHIFT e C= si accede al secondo codice di caratteri, chiamato "**MINUSCOLO/MAIUSCOLO**".
Osservate allora sullo schermo che:

la A è trasformata in a
la ↑ è trasformata in A
il carattere grafico ┌ non cambia.

Il ritorno in modo "MAIUSCOLO/GRAFICO" si effettua ugualmente premendo SHIFT e C= simultaneamente.

SHIFT LOCK

Il tasto SHIFT LOCK blocca la tastiera.

- in "**modo grafico**" se ci si trova in "MAIUSCOLO/GRAFICO"
- in "**maiuscolo**" se ci si trova in "MINUSCOLO/MAIUSCOLO".

Per tornare in modo normale, occorre premere di nuovo SHIFT LOCK.

Nota: i tasti carattere non sono a ripetizione.

Premendo la barra "SPAZIO" cancellate il carattere che si trova sotto il cursore.
Contrariamente ai tasti carattere, la barra "SPAZIO" è a ripetizione.

Tasti funzionali dello schermo

Questi tasti servono a posizionarsi sullo schermo:

TASTI	FUNZIONI (SENZA SHIFT)	FUNZIONI (CON SHIFT O C=)
↑ **CRSR** ↓	Sposta il cursore di una linea verso il basso	Sposta il cursore di una linea verso l'alto
← **CRSR** →	Sposta il cursore di un carattere verso destra	Sposta il cursore di un carattere verso sinistra
CLR HOME	Posiziona il cursore alto a destra dello schermo	Cancella lo schermo e posiziona il cursore in alto a sinistra
INST DEL	Cancella il carattere che si trova a sinistra del cursore ed avvicina tutta la parte destra della linea di un carattere	Crea uno spazio sotto il cursore spostando la parte destra della linea di un carattere

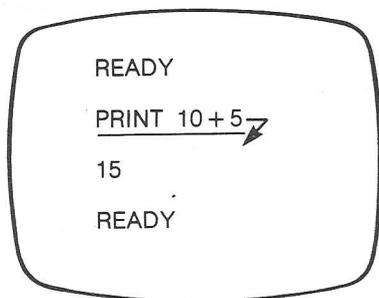
Non premete un tasto "cursore" in uno spazio creato per inserimento, poiché fareste apparire un carattere in "reverse". Per spostarvi all'interno dei "blanks" utilizzate la barra spaziatrice.
Vi raccomandiamo di imparare l'utilizzo di questi tasti prima di proseguire. (Studieremo gli altri tasti in seguito).
Non preoccupatevi dei diversi messaggi di errore che possono comparire sullo schermo, in nessun caso rischiate di deteriorare il vostro computer.

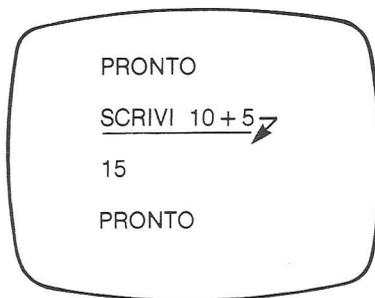
Il modo immediato (chiamato anche modo diretto)

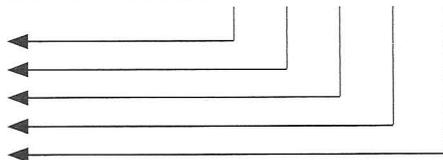
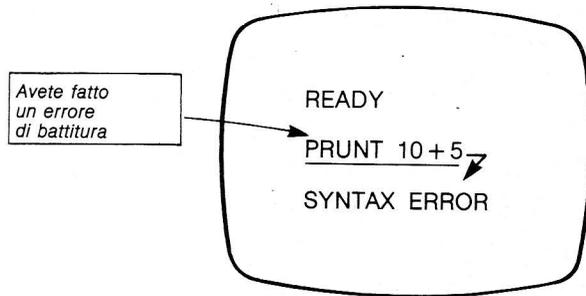
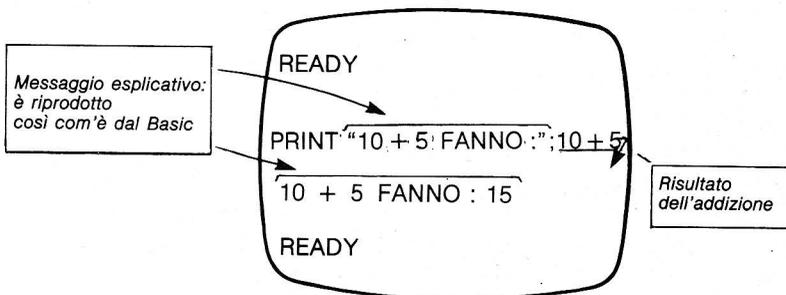
Per dialogare con il computer, sfortunatamente non basta conoscere bene la sua tastiera: bisogna anche parlare la sua lingua, il BASIC. Non cominciate a preoccuparvi, vedrete com'è semplice!
Un breve periodo di apprendimento vi basterà per diventare perfettamente bilingue.

Detto questo, per iniziare la conversazione vi basterà battere un'istruzione, che non dimenticherete di convalidare per mezzo di < RETURN >. Il computer allora vi risponderà IMMEDIATAMENTE.
Cominciamo:

Versione inglese

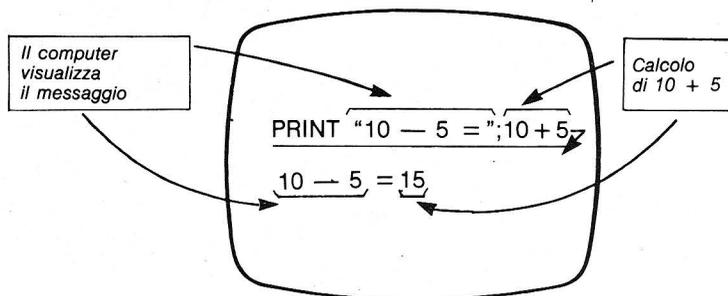
— Voi: PRINT 10 + 5
— Lui: 15

Versione italiana

— Voi: SCRIVI 10 + 5
— Lui: 15

Guardiamo più da vicino lo schermo in versione B A S I C

(per principianti) Beginner's
(tutti gli usi) All-purpose
Symbolic
Instruction
Code

Come per tutte le lingue, esiste in BASIC una sintassi che dovete rispettare. In caso contrario, il computer non comprenderà il messaggio che gli trasmettete. Ve lo segnala facendo apparire il messaggio SYNTAX ERROR (errore di sintassi) ed attende che voi battiate di nuovo, senza errori, la vostra istruzione.

Immaginiamo che vi siate accorti di questo imperdonabile (parliamo dal punto di vista della macchina) errore di battitura prima di avere premuto il tasto < RETURN >.
Cosa avreste potuto fare per rimediare a questa dimenticanza?
Come avrete notato, il computer si contenta di darvi una risposta senza fare lunghi discorsi.
Sta a voi curarvi di migliorare la presentazione della risposta.
Per fare questo, vi basta scrivere un MESSAGGIO posto TRA VIRGOLETTE.

ATTENZIONE ALLE VIRGOLETTE!

Sono le virgolette ad avvertire il computer che si tratta di un messaggio. Questo messaggio è del tutto arbitrario.
Avreste potuto tranquillamente battere:

```
PRINT "10 — 5 =";10+5
```

Il computer memorizza il messaggio posto tra virgolette e lo fa apparire testualmente.

Questo vi permette di constatare la sua fedeltà, certo, ma anche la sua mancanza "d'intelligenza".

In breve

• *Il modo **IMMEDIATO** (o **DIRETTO**) vi permette di conoscere "immediatamente" l'effetto di un'istruzione.*

• *L'istruzione **PRINT (SCRIVI)** è usata per visualizzare sullo schermo i dati a cui si riferisce.*

• *Il messaggio **SYNTAX ERROR** indica che il computer non ha compreso ciò che gli avete domandato. Impotente, vi domanda di ribattere l'istruzione senza sbagliarvi nella formulazione.*

• *Le **VIRGOLETTE** "..." indicano alla macchina che deve **SOLAMENTE** riprodurre il messaggio che queste racchiudono.*

• *Il **PUNTO E VIRGOLA** (;) in una istruzione **PRINT** indica alla macchina che deve scrivere il dato successivo, al seguito del precedente.*

Memorizzazione dei valori: variabili

L'istruzione PRINT, per quanto importante sia, non ci mostra granché delle possibilità della macchina.

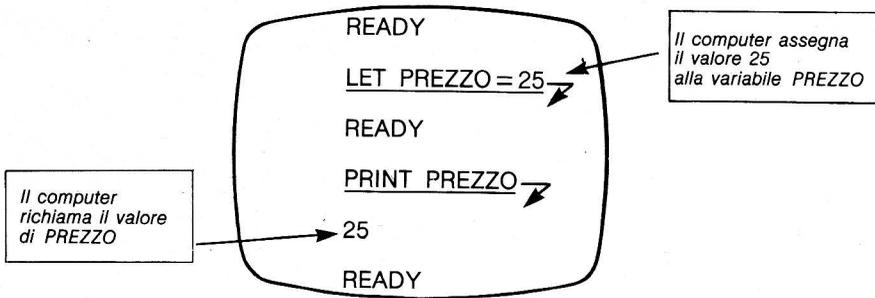
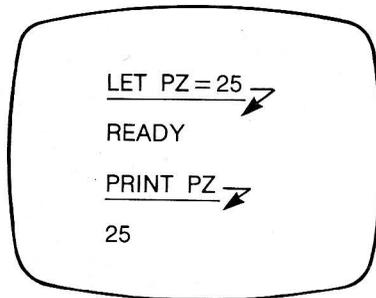
L'istruzione LET variabile = valore

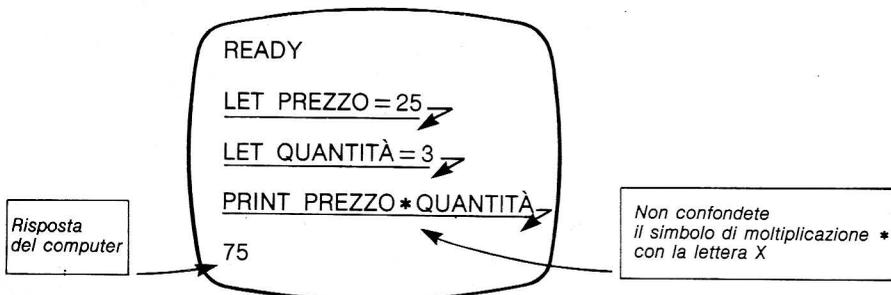
Questa istruzione permette di conservare in memoria dei valori che verranno utilizzati in seguito (per esempio, per compiere dei calcoli).

Per comprendere il significato di "variabile", paragoniamola ad una casella (d'ora in poi la simbolizzeremo così) all'interno della quale collochiamo un valore. Diamole poi un nome per IDENTIFICARLA.
Così, per assegnare il valore 25 ad una variabile che chiamiamo PREZZO, scriviamo:

```
LET PREZZO=25
```

Il BASIC colloca allora il valore 25 nella casella PREZZO.

Il computer richiama il valore di PREZZO. Per visualizzare questo valore, è sufficiente battere PRINT PREZZO.

Il nome dato alla variabile (qui PREZZO) è arbitrario; per verificarlo, provate a fare:

Tuttavia è importante che il nome della variabile sia il più mnemonico possibile.

Calcoliamo ora il totale del prodotto di 25 lire per una quantità di 3.
Per fare questo, utilizziamo 2 variabili (2 caselle).

I nomi delle variabili "PREZZO" e "QUANTITÀ" possono essere rimpiazzati da "PZ" e "QT".

```
LET PZ=25
LET QT=3
PRINT PZ*QT
75
```

Otteniamo lo stesso risultato scrivendo:

```
LET PZ=25
LET QT=3
LET T=PZ*QT
PRINT T
75
```

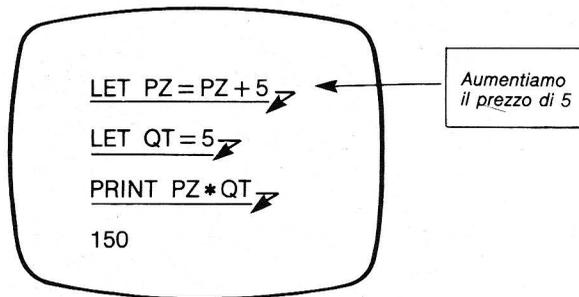
Vediamo ora come cambiare il valore della variabile QT. Per darle il valore 5, è sufficiente fare:

```
LET QT=5
```

(la variabile PZ conserva il suo vecchio valore).

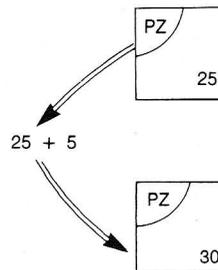
Naturalmente, il computer "dimentica" il vecchio valore di QT.

Immaginiamo ora che il prezzo aumenti di 5 lire.
Potremmo certamente fare: LET PZ = 30.

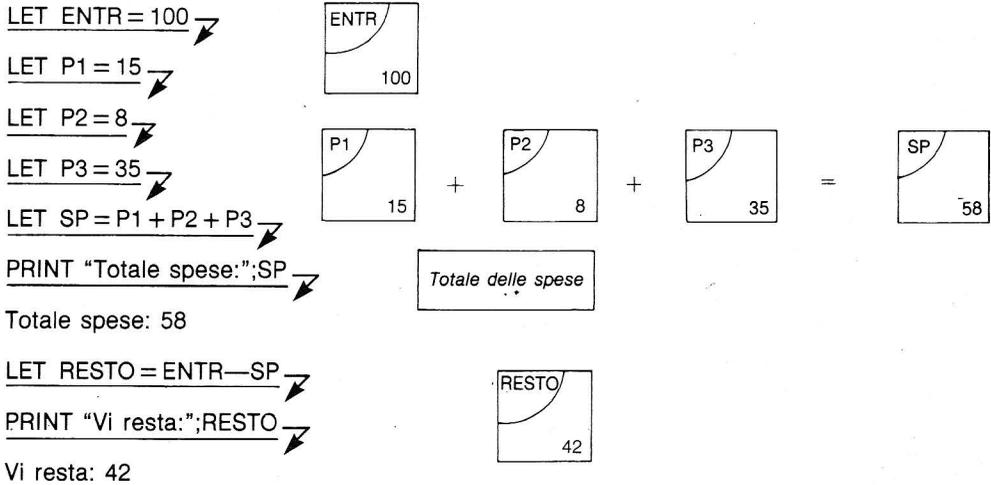
Ma possiamo anche fare:

LET PZ = PZ + 5 non deve essere interpretata come una identità algebrica. Il computer interpreta questa addizione nel modo seguente:

1. Prende il contenuto della casella PZ.

2. Aggiunge 5.

3. Colloca il totale nella casella PZ.

Esempi

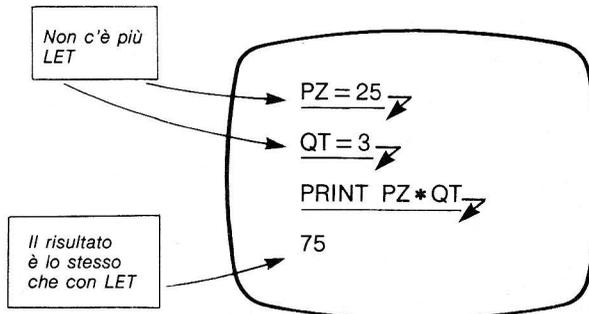
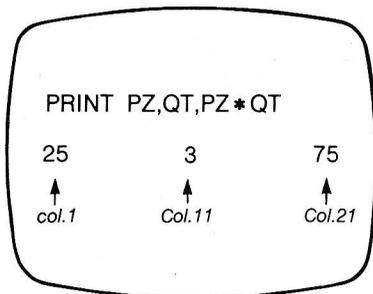
Calcoliamo una data di nascita in funzione dell'età:

Calcoliamo ora il totale di alcune spese (P1,P2,P3) e l'importo rimanente:

Per verificare i valori di P1, P2, P3, battiamo:

```
PRINT P1;P2,P3
15      8      25
```

Se abbiamo commesso un errore nel prezzo di P2, ci basta fare:

```
LET P2 = 18
LET SP = P1 + P2 + P3
PRINT "Totale spese:";SP
Totale spese: 68
```

Per assegnare un valore ad una variabile, abbiamo utilizzato l'istruzione LET. Oggi, per la maggior parte dei BASIC, non è più obbligatoria.

Per visualizzare sulla stessa linea PREZZO, QUANTITÀ e PREZZO * QUANTITÀ, separiamo le variabili per mezzo di virgole.

I risultati sono visualizzati nelle colonne 1, 11 e 21 (di 10 in 10).
Il punto e virgola impedisce il salto di linea ma non incolonna i risultati, che vengono visualizzati l'uno di seguito all'altro, separati da uno spazio.

```
PRINT PZ;QT;PZ*QT

25  3  75
```

Esistono tre tipi di variabili:

- **VARIABILE NUMERICA**
Il suo contenuto può essere solo numerico, positivo o negativo, con o senza decimali.

- **VARIABILE NUMERICA INTERA**
Il nome di questa variabile è completato dal simbolo % (per esempio RE%). Il suo contenuto può essere solo un numero intero compreso fra + 32.765 e —32.765.

- **VARIABILE STRINGA DI CARATTERI**
Il nome di questa variabile è completato dal simbolo $ (per esempio NM$). Il suo contenuto può essere numerico, alfabetico o grafico.
Il valore da assegnare a questa variabile deve essere scritto **obbligatoriamente** tra virgolette (per esempio: NM$ = "GIUSEPPE").
Parleremo più dettagliatamente di questa variabile nel capitolo "stringhe di caratteri".

Note sui nomi delle variabili

Il primo carattere deve obbligatoriamente essere una lettera. Gli altri caratteri possono essere indifferentemente lettere o cifre.

A2% e A2$ sono accettate
2A% e 2A$ sono rifiutate

Per il BASIC, PREZZO e PR sono la stessa variabile.

ATTENZIONE! Il Commodore 64 non accetta nomi di variabili contenenti un comando BASIC.
Le variabili **TO, TI, TI$** e **ST** sono rifiutate:

TO è un comando BASIC
TI
TI$ } danno il tempo trascorso dall'accensione o dall'inizializzazione
TI in 100° di secondo
TI$ in ore, minuti, secondi secondo la forma "HH MM SS"
ST è una variabile di sistema.

In breve

- *Una **VARIABILE** è una "casella" identificabile da un nome mnemonico, nella quale si colloca un valore qualsiasi.*

- *Si può aggiungere un valore ad una variabile X facendo:*

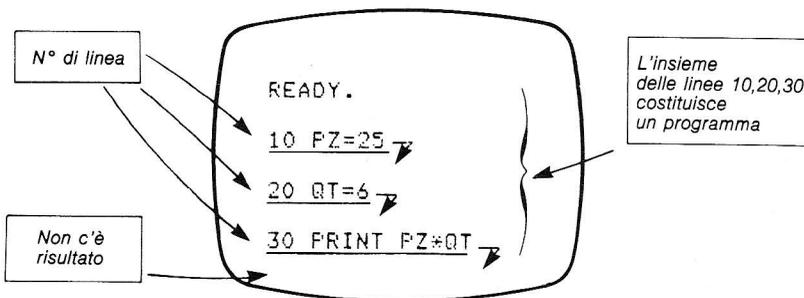
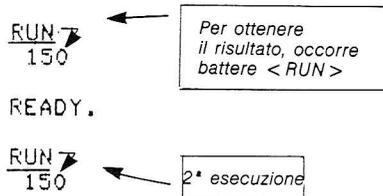
  LET X = X + valore (o X = X + valore)

- *La VIRGOLA permette di visualizzare i risultati nelle colonne 1, 11, 21,...*

- *Il PUNTO E VIRGOLA non incolonna i risultati, ma li visualizza l'uno di seguito all'altro.*

Memorizzazione delle istruzioni: il programma

Invece di battere più volte la stessa sequenza di istruzioni, la possiamo memorizzare.

Per memorizzare una sequenza di istruzioni, le battiamo assegnando loro un NUMERO DI LINEA (10, 20, 30, ...).
Questa sequenza di istruzioni si chiama PROGRAMMA.

Possiamo constatare che non c'è risultato. **Per ottenere un risultato, bisogna richiedere l'ESECUZIONE DEL PROGRAMMA con l'ordine RUN.**

Il BASIC esegue le istruzioni ad una ad una, seguendo la numerazione.
L'esecuzione del programma può essere richiesta più volte.
Il BASIC le esegue automaticamente.

Nota: le righe possono venire inserite anche senza seguire l'ordine della numerazione.

Modifica ad una riga

La cosa più semplice è ribatterla:

Aggiunta di una riga

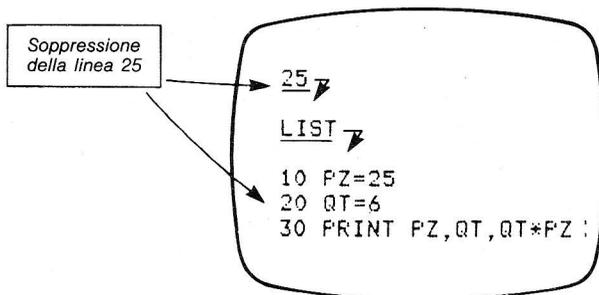
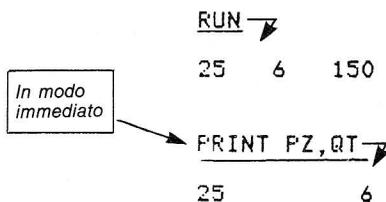
L'abitudine vuole che la numerazione si faccia di 10 in 10.
Per inserire una nuova riga (in 25, per esempio), basta fare:

Lista di un programma

Il comando LIST pemette di visualizzare il programma presente in memoria (o una parte di esso) ogni volta che lo desideriamo.

LIST 10 → visualizza la linea 10
LIST — 30 → visualizza dalla linea 10 alla 30
LIST 20 — → visualizza dalla linea 20 alla fine
LIST 20-40 → visualizza dalla linea 20 alla 40

Eliminazione di una riga

Battendo il numero di riga seguito da < RETURN >, la riga è cancellata.

Alla fine dell'esecuzione del programma (dopo RUN), le variabili PZ e QT hanno conservato il loro valore. Lo si può constatare battendo direttamente (cioè senza numero di riga): PRINT PZ, QT.

Per conoscere l'effetto di una istruzione, può essere più rapido batterla direttamente invece che consultare un manuale.
Così, per la messa a punto, il modo diretto è utilissimo.
È quindi essenziale acquisire l'abitudine a farne uso.

Eliminazione di un programma

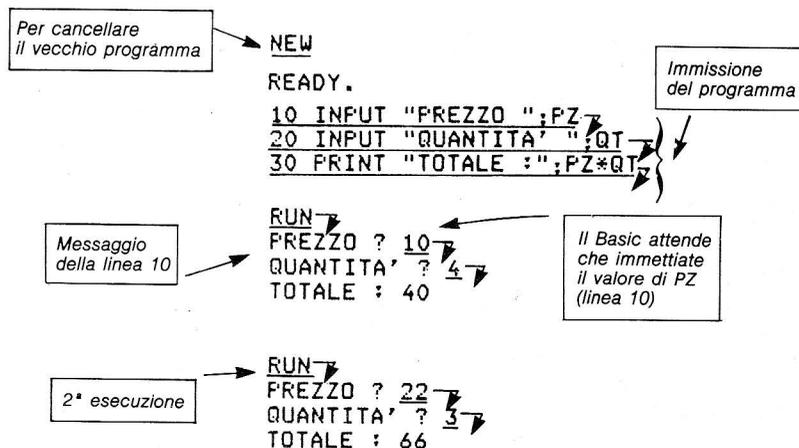
Utilizzare il comando "NEW".

In breve

- *Un **PROGRAMMA** è una sequenza di istruzioni numerate.*
- *La numerazione si fa generalmente di 10 in 10 per consentire l'inserimento di nuove righe.*
- *Il comando **RUN** lancia l'esecuzione di un programma.*
- *Il comando **LIST** visualizza sullo schermo il programma presente in memoria.*
- *In modo **"MINUSCOLO/MAIUSCOLO"**, battete i comandi BASIC in minuscolo.*

Per immettere valori durante l'esecuzione: INPUT "Messaggio?"; variabile

L'istruzione INPUT (IMMETTERE) permetterà di fornire dei valori al programma DURANTE l'ESECUZIONE attraverso la TASTIERA.

Quando il BASIC incontra un'istruzione INPUT durante l'esecuzione di un programma, visualizza sullo schermo il messaggio che figura nell'istruzione INPUT (nell'esempio Prezzo?) ed aspetta che l'operatore immetta da tastiera il valore della variabile specificata nell'istruzione INPUT (nell'es., PZ).
Quando l'operatore ha immesso il valore seguito da < RETURN >, l'esecuzione del programma prosegue alla riga successiva.

Note

- Al momento dell'esecuzione, il Commodore 64 aggiunge automaticamente un punto interrogativo dopo il messaggio.
- Il comand NEW permette di cancellare il vecchio programma in memoria (allo stesso modo le nuove righe cancellerebbero le vecchie).

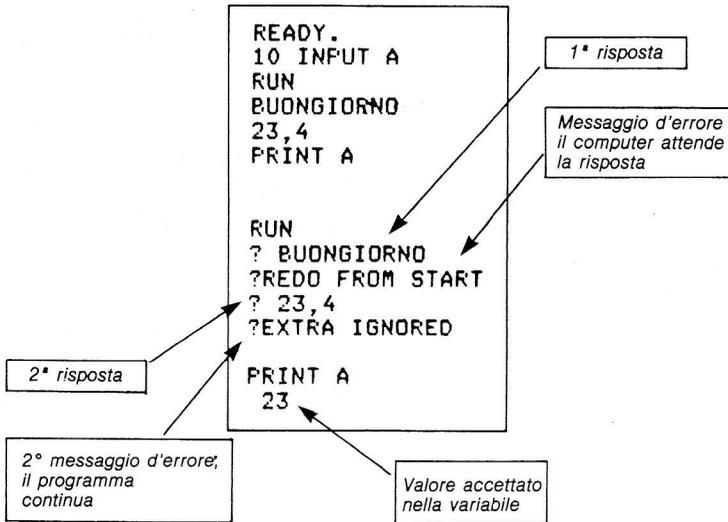
L'istruzione INPUT è molto utile poiché permette di eseguire il programma con valori differenti senza dover modificare il programma stesso.

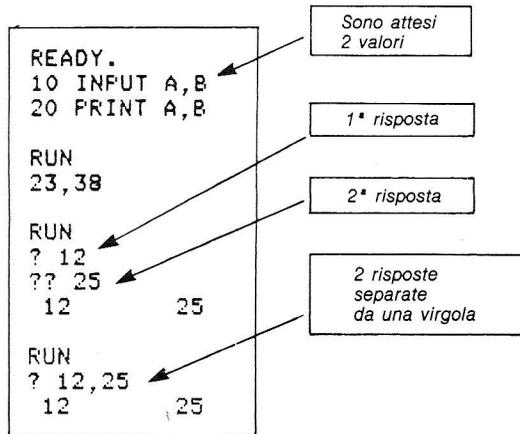
Esempio

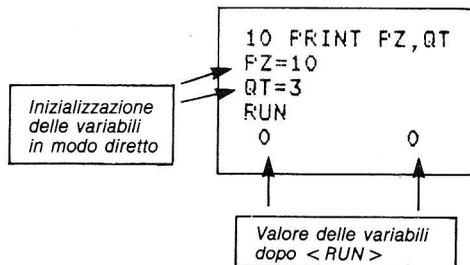
Calcoliamo il consumo di un veicolo.

```
10 INPUT "QUANTI LITRI ";L
20 INPUT "QUANTI KM ";KM
30 :
40 C=(L/KM)*100
50 :
60 PRINT "CONSUMO PER 100 KM :";C
70 :
80 GOTO 10


RUN
QUANTI LITRI ? 40
QUANTI KM ? 520
CONSUMO PER 100 KM : 7.69230769
```

Durante l'esecuzione di un'istruzione INPUT, possono apparire sullo schermo due messaggi di errore:

- **REDO FROM START:** il computer vi chiede di ricominciare, poiché nella vostra risposta uno o più caratteri non erano numerici.
- **EXTRA IGNORED:** avete battuto una virgola invece di un punto in numero decimale; il computer quindi ha accettato solamente le cifre prima della virgola. Ricominciate!

In un'istruzione INPUT possono essere specificate due variabili. In questo caso, i due valori che immettete durante l'esecuzione devono essere separati da una virgola.

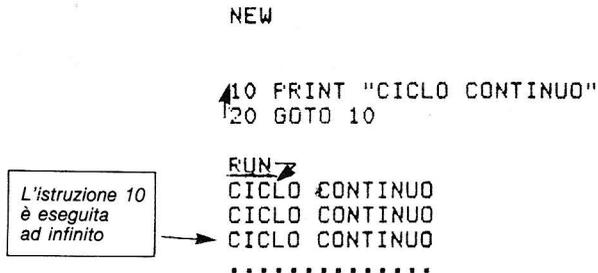
Se immettete un solo valore, il computer vi mostra due punti interrogativi ed attende che immettiate il secondo valore.

Note

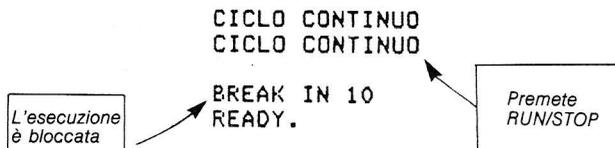
- Una modifica nel programma annulla il valore delle variabili.
- RUN inizializza il valore delle variabili a 0.

In breve

- *L'istruzione **INPUT** permette di introdurre i valori delle variabili soltanto al **MOMENTO** dell'**ESECUZIONE** del **PROGRAMMA**.*

Loop:
GOTO numero di riga

L'istruzione "GOTO n. riga" permette di eseguire più volte la stessa sequenza di istruzioni.

Mostriamo in un esempio (senza utilità pratica) il meccanismo di GOTO:

Il BASIC esegue l'istruzione 10 una prima volta, poi passa ad eseguire l'istruzione 20. Questa istruzione "20 GOTO 10" significa "ANDARE ALLA RIGA 10". L'istruzione 10 è quindi eseguita nuovamente.

Naturalmente, questo programma non arresta mai... a meno che voi premiate il tasto RUN/STOP.

Utilizziamo questa istruzione GOTO in un contesto più vicino ai problemi reali. Supponiamo di dover compiere più volte la somma di due numeri.

```
10 INPUT "PRIMO NUMERO ";N1
20 INPUT "SECONDO NUMERO ";N2
30 PRINT "SOMMA :";N1+N2
40 PRINT "PRODOTTO :";N1*N2
50 GOTO 10

RUN
PRIMO NUMERO ? 5
SECONDO NUMERO ? 3
SOMMA : 8
PRODOTTO : 15


PRIMO NUMERO   ? 8
SECONDO NUMERO ? 3
SOMMA : 11
PRODOTTO : 24


PRIMO NUMERO   ?
```

Si ritorna alla linea 10

Premete RUN/STOP e RESTORE

• Per fermare un programma in attesa su "INPUT", premete simultaneamente RUN/STOP e RESTORE.

Esempio

Per provocare uno slittamento di riga prima della visualizzazione di "sei nato nel:", aggiungete l'istruzione:

Nota: potete anche scrivere "25 ?".

Al momento dell'esecuzione il ? è sostituito dall'istruzione PRINT.

Istruzione REM

Non si tratta di una vera e propria istruzione, poiché non influenza lo svolgimento del programma.

Essa permette di introdurre delle note nel programma e di migliorarne la presentazione (REM deriva da REMark).

In breve

- *L'istruzione* ***GOTO n. riga (ANDARE A*** *n. riga) provoca il proseguimento del programma alla riga indicata.*

- ***REM*** *permette di introdurre dei commenti nel programma. È ignorata durante l'esecuzione.*

- *Al fine di migliorare la lettura di un programma, è consigliabile inserirvi dei commenti usando REM (o ').*

- ***RUN/STOP*** *permette di* ***BLOCCARE*** *l'esecuzione del programma.*

Come salvare un programma

SAVE

Il comando **SAVE "NOME del programma"** permette di memorizzare su cassetta il programma presente in memoria.

SAVE "ANNUARIO"

Il programma in memoria sarà denominato "ANNUARIO" e memorizzato su cassetta.

Naturalmente sulla stessa cassetta si possono salvare più programmi. Fate però attenzione nel trovare l'inizio e la fine di ogni programma, usando il contagiri del vostro registratore, per non fare accavallare due programmi. Infatti, la parte finale del primo programma sarebbe distrutta dall'inizio del secondo.

VERIFY

Questo comando permette di **"VERIFICARE"** che il programma sia correttamente salvato.

Battete **VERIFY "ANNUARIO":**
il computer cercherà sulla cassetta già riavvolta il programma ANNUARIO. Una volta trovato, verifica byte per byte se è conforme al programma presente in memoria centrale.

Nota: se il nome del programma da verificare non è specificato, il computer verifica il primo programma che trova sulla cassetta.

LOAD

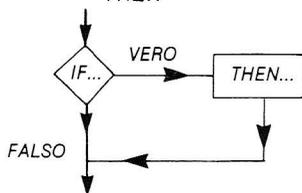
Il comando **LOAD "NOME del programma"** carica in memoria il programma "NOME del programma" presente sulla cassetta.
L'istruzione NEW è eseguita automaticamente: i dati precedenti sono cancellati.
Il comando LOAD utilizzato da solo carica in memoria il primo programma presente sulla cassetta.
Premendo i tasti "SHIFT/RUN STOP" caricate in memoria il primo programma presente sulla cassetta e ne lanciate automaticamente l'esecuzione.

Tests:
IF condizione THEN istruzione
(SE condizione ALLORA istruzione)

L'istruzione IF... THEN... permette di verificare una condizione e di eseguire una o più istruzioni se essa è vera.

```
10 INPUT "BATTI UN NUMERO ";N1
20 IF (N1>10) THEN PRINT "HAI BATTUTO UN NUMERO SUPERIORE A 10"
30 PRINT "SEGUITO"
40 GOTO 10


RUN
BATTI UN NUMERO ? 15
HAI BATTUTO UN NUMERO SUPERRIORE A 10
SEGUITO


BATTI UN NUMERO ? 8
SEGUITO
```


L'istruzione 20 significa:

```
20 SE N1 E' SUPERIORE A 10 ALLORA SCRIVI "HAI BATTUTO UN .."   ..."
```

Sia che l'istruzione dopo THEN sia eseguita o meno, il programma prosegue con la riga che segue IF... THEN...

È in questa istruzione (come in GOTO) che risiede tutta la potenza del computer, poiché essa rende possibile eseguire sequenze di istruzioni solo in determinati casi.

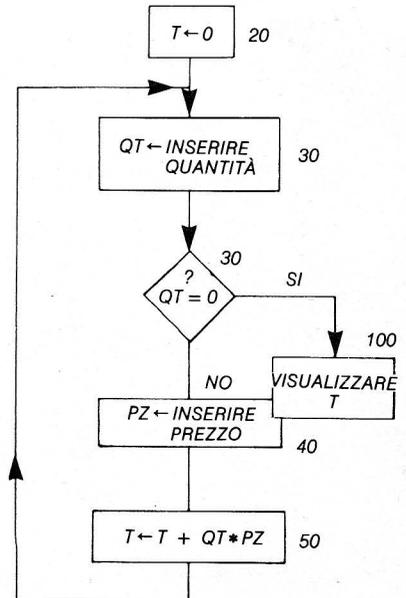
Nota: per eseguire più istruzioni, basta separarle con il carattere ":" (due punti).

Le diverse **operazioni di confronto** sono:

= uguaglianza > maggiore di ⇒ maggiore o uguale a
< minore di < > diverso da ⇐ minore o uguale a

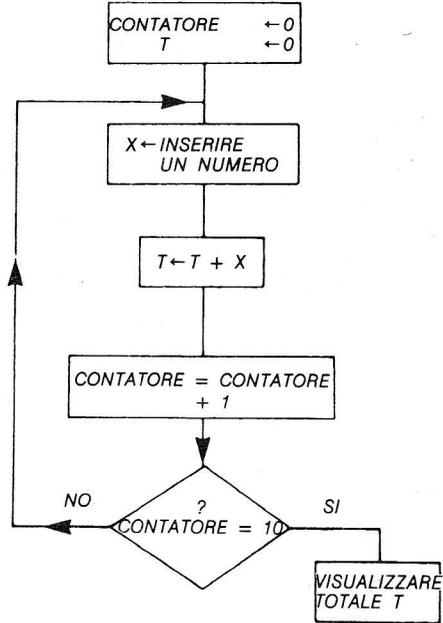
Uscita da un loop

L'istruzione IF... THEN... permette di "uscire" da un loop.

Il seguente programma effettua la totalizzazione degli acquisti.

Quando tutti gli acquisti sono stati sommati, rispondiamo 0 alla domanda "Quantità?".

La comparazione del valore 0 alla riga 30 permette di uscire dal loop per visualizzare il totale.

Uscita da un loop per mezzo di un contatore

Un altro metodo per uscire da un loop consiste nel contare il numero dei passaggi nel loop usando una variabile (CONTATORE nell'esempio). Una comparazione del valore di CONTATORE permette di uscire dal loop.

Se non si conosce il numero dei valori da aggiungere basta fare:

Test di condizioni multiple

Possono essere realizzate delle comparazioni multiple per mezzo degli OPERATORI LOGICI AND e OR.

Condizione 1 AND condizione 2: Le istruzioni dopo THEN sono eseguite solo se la condizione 1 E la condizione 2 sono vere.

```
10 INPUT "BATTI UN NUMERO ";N
20 IF N>0 AND N<10 THEN PRINT "IL NUMERO E' MAGGIORE DI 0 E MINORE DI 10"
30 GOTO 10
```

```
BATTI UN NUMERO ? 5
IL NUMERO E' MAGGIORE DI 0 E MINORE DI 10

BATTI UN NUMERO ? 13

BATTI UN NUMERO ?
```

Condizione 1 AND condizione 2: Le istruzioni dopo THEN sono eseguite se una delle due condizioni è vera (o lo sono entrambe).

```
10 INPUT "BATTI DUE NUMERI X,Y ";X,Y
20 IF (X>0) OR (Y>0) THEN PRINT "X O Y E' POSITIVO (O ENTRAMBI)"
30 GOTO 10
```

```
RUN
BATTI DUE NUMERI X,Y ? 12,4
X O Y E' POSITIVO (O ENTRAMBI)

BATTI DUE NUMERI X,Y ? -4,-1

BATTI DUE NUMERI X,Y ?
```

Loop automatico: FOR... NEXT

Dobbiamo visualizzare i numeri 1, 2, 3 e i loro quadrati 1 * 1, 2 * 2, 3 * 3.

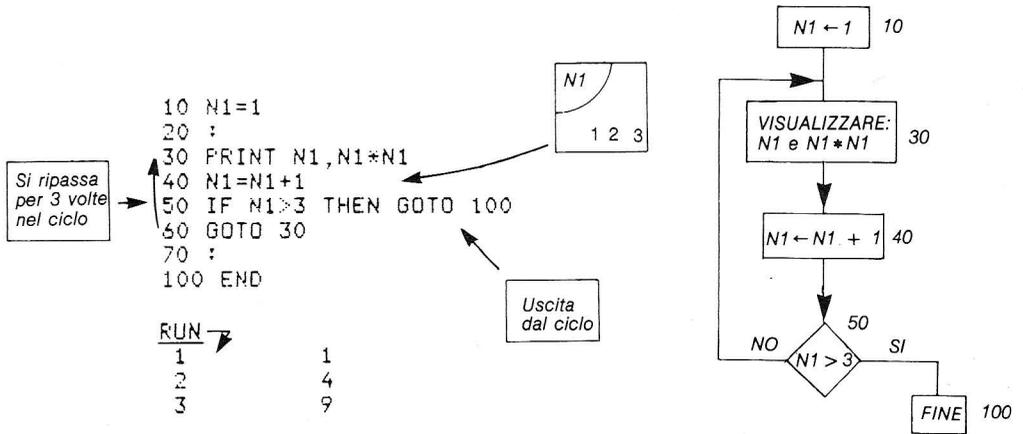
Prima soluzione: (senza GOTO e IF... THEN...)

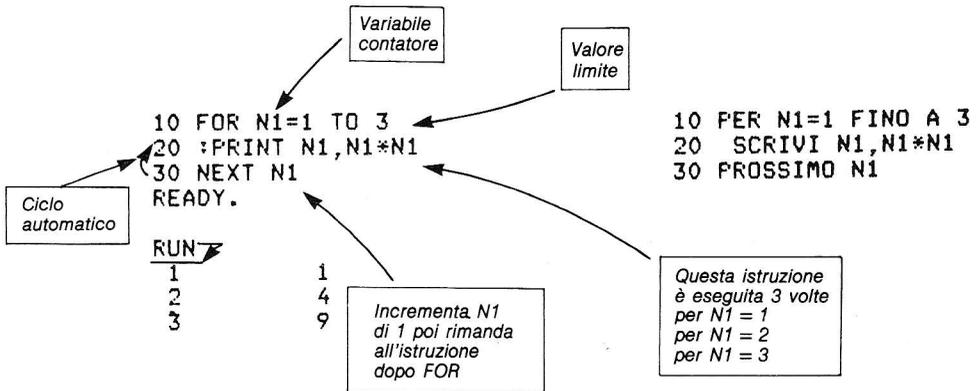
Per calcolare i quadrati dei numeri da 1 a 100, questo metodo risulterebbe lungo e fastidioso.

Seconda soluzione: (con GOTO e IF... THEN...)

In questo modo, invece di scrivere molte volte la sequenza:

N1 = N1 + 1
PRINT N1, N1 * N1

scriviamola una sola volta.
Per eseguirla più volte, utilizzeremo un'istruzione di allacciamento GOTO, e, per uscire dal loop, compareremo il valore di N1. Una volta che N1 è diventato superiore a 3, faremo fermare il programma.

Terza soluzione: (loop automatico FOR... NEXT...)

Semplifichiamo il programma precedente con un loop automatico: FOR... NEXT...

Come si comporta il programma?

— La riga 10 colloca in memoria il "valore limite" 3 ed assegna alla "variabile contatore" N1 il valore 1.
— La riga 20 è eseguita con N1 = 1.
— La riga "30 NEXT N1" aumenta N1 di uno e controlla se N1 è MINORE od UGUALE al "valore limite" 3.

Se N1 è minore od uguale a 3, la riga seguente FOR (la 20) è eseguita nuovamente con N1 = 2.
In questo modo, la riga 20 è eseguita 3 volte con N1 = 1, 2, 3.

Esempi

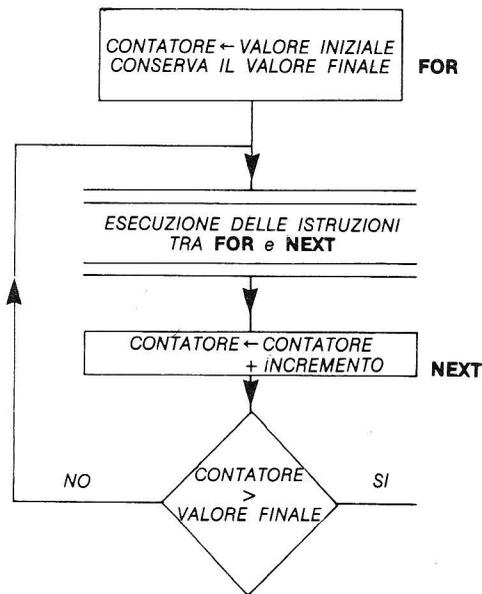
Per visualizzare 5 linee di asterischi:

```
10 FOR N=1 TO 5
20 :PRINT "*****"
30 NEXT N
READY.
RUN
*****
*****
*****
*****
*****
```

Per visualizzare i quadrati dei numeri da 4 a 6:

```
10 FOR N1=4 TO 6
20 :PRINT N1,N1*N1
30 NEXT N1
40 PRINT "IL LOOP E' TERMINATO"

RUN
 4        16
 5        25
 6        36
IL LOOP E' TERMINATO
```

Sintassi completa del ciclo FOR

Un passo di esecuzione può essere specificato come STEP:
La specifica "STEP valore dell'incremento" indica al computer che deve aumentare la variabile contatore del valore indicato ad ogni esecuzione del loop.
Omettendo la specifica STEP il contatore viene incrementato automaticamente di 1 ad ogni passaggio. Dunque è la medesima cosa non specificare il valore dell'incremento od assegnargli il valore 1.

FOR contatore = valore iniziale TO valore finale STEP incremento
istruzione 1
istruzione 2
. . . .
NEXT contatore

Quando l'istruzione FOR è eseguita, il BASIC assegna alla "variabile contatore" il "valore iniziale" specificato e conserva in memoria il "valore limite" indicato.
Tutte le istruzioni tra FOR e NEXT sono inizialmente eseguite con contatore = valore iniziale.
L'esecuzione dell'istruzione NEXT aumenta il valore del "contatore" dell'incremento indicato.

- Se il valore del "contatore" è minore del "valore limite", le istruzioni tra FOR e NEXT sono eseguite nuovamente con il nuovo valore del "contatore".
- Se il valore del "contatore" ha raggiunto il valore finale l'esecuzione del loop si interrompe ed il programma prosegue con la riga che segue NEXT.

Esempi

STEP positivo:

```
10 FOR C=1 TO 5 STEP 2
20 :PRINT C;
30 NEXT C
40 PRINT "E' FINITO"

RUN
 1  3  5 E' FINITO
```

Tabella dei seni:

```
10 FOR ANG=0 TO 3.14 STEP 3.14/10
20 :PRINT ANG,SIN(ANG)
30 NEXT ANG

RUN
 0            0
 .314         .30886552
 .628         .587527526
 .942         .808736061
```

STEP negativo:

```
10 FOR C=3 TO 1 STEP-1
20 :PRINT C;
30 NEXT C

RUN
 3  2  1
```

Un ciclo FOR è eseguito almeno una volta, anche se il "valore limite" è minore del "valore iniziale". (La comparazione di fine è fatta al momento dell'esecuzione di NEXT).

```
10 D=3:F=1
20 :
30 FOR C=D TO F
40 :PRINT C
50 NEXT C

RUN
 3
```


Dobbiamo quindi prevedere una comparazione.

```
35 IF C>F THEN 50
```

Errori:

```
10 FOR I=1 TO 5
20 :PRINT I
30 NEXT J
READY.
RUN
 1

?NEXT WITHOUT FOR  ERROR IN 30
```


Esempio

Tabellina di moltiplicazione per 8:

Questo ciclo FOR visualizza la tabellina di moltiplicazione per 8.

```
30 FOR N1=1 TO 10
40 :PRINT N1;"X";8;"=";N1*8
50 NEXT N1
READY.
RUN
 1 X 8 = 8
 2 X 8 = 16
 3 X 8 = 24
 4 X 8 = 32
 5 X 8 = 40
 6 X 8 = 48
 7 X 8 = 56
 8 X 8 = 64
 9 X 8 = 72
 10 X 8 = 80
```

Senza ciclo, si dovrebbe scrivere:

```
10 PRINT 1;"X";8;"=";1*8
20 PRINT 2;"X";8;"=";2*8
30 PRINT 3;"X";8;"=";3*8
40 PRINT 4;"X";8;"=";4*8
```

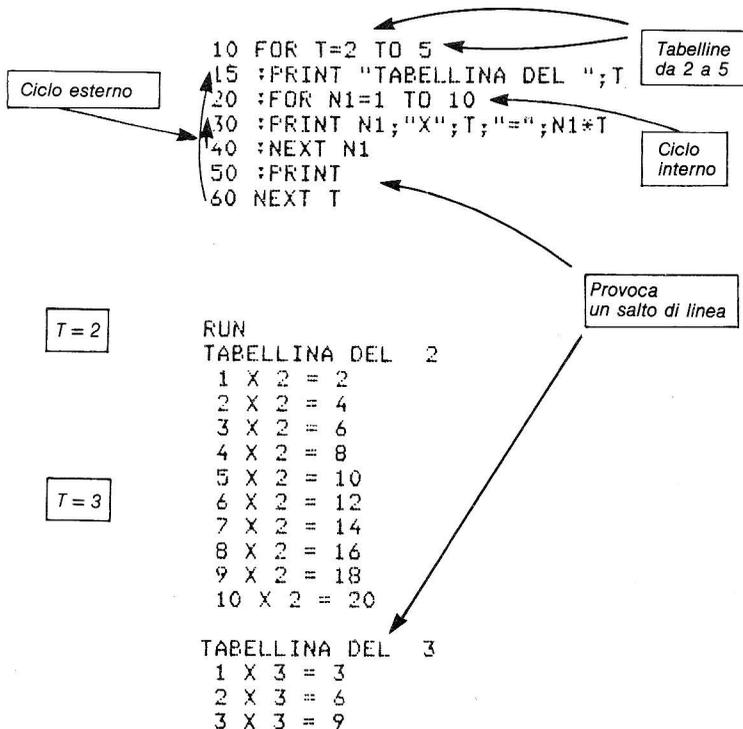
Questo sarebbe senz'altro troppo scomodo.

Per generalizzare questo programma a tutte le tabelline di moltiplicazione:

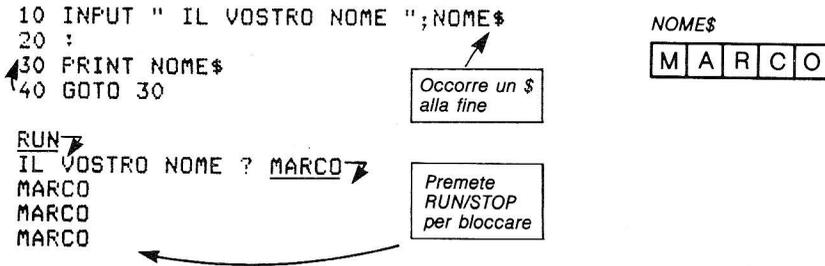
```
10 INPUT "QUALE TABELLINA ";T
20 :
30 FOR N1=1 TO 10
40 :PRINT N1;"X";T;"=";N1*T
50 NEXT N1
60 GOTO 10
```

```
RUN
QUALE TABELLINA ? 9
 1 X 9 = 9
 2 X 9 = 18
 3 X 9 = 27
 4 X 9 = 36
```

Loop ad incastro

Molti cicli FOR possono essere "inscatolati".
Il seguente programma visualizza le tabelline di moltiplicazione da 2 a 5.
Il loop "interno" è eseguito una prima volta con T = 2, poi con T = 3, ecc.

Le stringhe di caratteri

Esistono variabili chiamate "stringhe di caratteri". Si distinguono dalle variabili numeriche per la presenza del simbolo $ alla fine nome.

Queste variabili sono utilizzate nella maggioranza delle applicazioni (gestioni, istruzione, giochi).

Per impedire il salto di una linea dopo la stampa del nome, basta aggiungere un punto e virgola dopo NOME$.

```
30 PRINT NOM$;

RUN
IL VOSTRO NOME ? MARCO
MARCOMARCOMARCOMARCOMARCOMARCOMARCOMAR
```

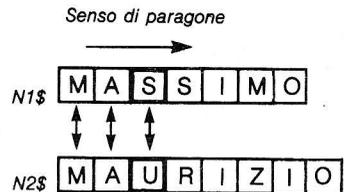
L'assegnazione del valore "MARCO" a NOME$ necessita la presenza delle virgolette. In caso contrario, "MARCO" sarà considerato come un nome di variabile.

La lunghezza delle stringhe è limitata a 255 caratteri.

• I caratteri possono essere anche cifre o caratteri grafici.

Comparazione di stringhe

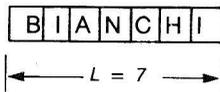
La comparazione di stringhe è effettuata dal BASIC carattere per carattere, da sinistra a destra.

Se un carattere di una stringa è diverso dal suo corrispondente, la comparazione si interrompe.

Concatenazione di stringhe

La concatenazione (o addizione) **di stringhe si compie per mezzo dell'operatore** "+" (come per le variabili numeriche).

Lunghezza di una stringa: LEN (stringa)

LEN (stringa) dà il numero dei caratteri di una stringa.

```
10 INPUT "IL VOSTRO COGNOME";COGN$
20 L=LEN(COGN$)
30 PRINT "IL VOSTRO COGNOME E' DI";L;"LETTERE"

RUN
IL VOSTRO COGNOME ? BIANCHI
IL VOSTRO COGNOME E' DI 7 LETTERE
```

COGN$

B	I	A	N	C	H	I

L = 7

La lunghezza di una stringa può essere variata durante l'esecuzione del programma. La lunghezza massima non deve essere assegnata all'inizio del programma.

LEFT$ (stringa, lunghezza da prendere)

Dà i carattere di sinistra di una stringa.

```
10 X$="64 PER TUTTI"
20 Y$=LEFT$(X$,2)
30 PRINT Y$

RUN
64
```


Per esempio:

```
10 INPUT "IL VOSTRO COGNOME";COGN$
20 FOR L=1 TO LEN(COGN$)
30 : PRINT LEFT$(NOM$,L)
40 NEXT L

RUN
IL VOSTRO COGNOME ? BIANCHI
B
BI
BIA
BIAN
BIANC
BIANCH
BIANCHI
```

RIGHT$ (stringa, lunghezza da prendere)

Dà i caratteri di destra di una stringa.

```
10 X$="64 PER TUTTI"
20 Y$=RIGHT$(X$,5)
30 PRINT Y$

RUN
TUTTI
```

Il seguente programma visualizza le combinazioni circolari dei caratteri di una stringa:

```
10 NOME$="BIANCHI"
20 PRINT NOME$
30 FOR N=1 TO LEN(NOME$)
40 : NOME$=RIGHT$(NOME$,1)+LEFT$(NOME$,LEN(NOME$)-1)
50 : PRINT NOME$
60 NEXT N
```

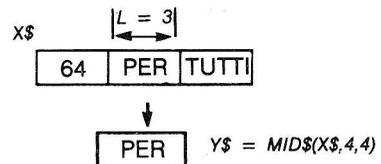
COGN$)M$

B	I	A	N	C	H	I

```
RUN
BIANCHI
IBIANCH
HIBIANC
CHIBIAN
NCHIBIA
ANCHIBI
IANCHIB
BIANCHI
```

MID$ (stringa, posizione iniziale, lunghezza da prendere)

Dà i caratteri centrali di una stringa.

Il programma seguente visualizza il nome a rovescio.

```
10 INPUT"IL VOSTRO COGNOME";COGN$
20 FOR P=1 TO LEN(COGN$) TO 1 STEP -1
30 : PRINT MID$(COGN$,P,1);
40 NEXT P


RUN
IL VOSTRO COGNOME ? BIANCHI
IHCNAIB
READY.
```

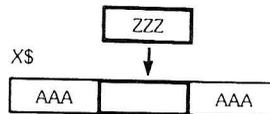
Questo scompone una frase:

```
10 INPUT"INSERITE UNA FRASE";FR$
20 :
30 FOR P=1 TO LEN(FR$)
40 : X$=MID$(FR$,P,1)
50 : IF X$ <>" " THEN Y$=Y$+X$
60 :IF X$=" " THEN PRINT Y$:Y$=" "
70 NEXT P
80 PRINT Y$

RUN
INSERITE UNA FRASE ? LA MAMMA HA FATTO I GNOCCHI
LA
MAMMA
HA
FATTO
I
GNOCCHI
```

PER SOSTITUIRE UNA PARTE DI STRINGA

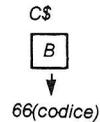
```
10 X$="AAAAAAAA"
20 P=4 : Y$="ZZZ"
30 GOSUB 100
40 PRINT X$
50 END
60 :
100 X$=LEFT$(X$,P-1)+Y$+RIGHT$(X$,LEN(X$)-P-LEN(Y$)+1)
110 RETURN


RUN
AAAZZZAA
```

ZZZ

X$

AAA | | AAA

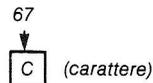
ASC (carattere)

Dà il codice di un carattere.

Tutti i caratteri sono rappresentati in linguaggio macchina sotto forma binaria. Il programmatore ha accesso a questi codici sotto forma decimale.

I codici dell'alfabeto (A, B, C,...Z) sono 65, 66, 67,...91.

ASC (stringa)

Dà il codice del primo carattere della stringa.

La stringa non deve essere vuota.

CHR$ (codice)

Fornisce il carattere identificato dal codice. Solo alcuni caratteri sono stampabili.

```
10 X=67
20 PRINT CHR$(X)

RUN
C
```

```
10 FOR C=65 TO 65+26
20 : PRINT CHR$(C);
30 NEXT C

RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Questo programma fa corrispondere ad ogni lettera dell'alfabeto la lettera successiva (per esempio, ad A viene fatta corrispondere B).

```
10 INPUT "IL VOSTRO COGNOME";COGN$
20 :
30 FOR P=1 TO LEN(COGN$)
40 : X=ASC(MID$(COGN$,P,1))
50 : PRINT CHR$(X+1);
60 NEXT P

RUN
IL VOSTRO COGNOME ? BIANCHI
CJBODIJ
```

Il cognome risultante

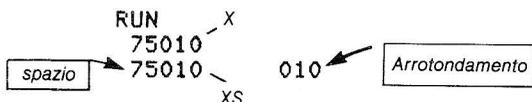
Alcuni caratteri servono per inviare comandi alle periferiche:

Alcuni codici ASCII

PRINT CHR$(10); provoca un salto di linea (senza ritorno a capo)
PRINT CHR$(13); provoca un ritorno a capo (senza salto di linea)
PRINT CHR$(147); cancella lo schermo

STR$(X)

Converte un numero sotto forma di stringa, permettendo così l'accesso a ciascuna cifra per mezzo delle funzioni LEFT$, RIGHT$, MID$.

```
10 X=75010
20 X$=STR$(X)
30 PRINT X
40 PRINT X$,RIGHT$(X$,3)
```

Notate la presenza di un carattere all'inizio della stringa. Esso corrisponde alla posizione del segno + (implicito). Il suo valore è 32.

VAL (stringa)

Dà il valore numerico di una stringa che inizia per cifre (o per uno spazio).

Se la stringa inizia con una lettera, il valore è nullo.

```
10 PREZZO$="13 MILIONI"
20 X=VAL(PREZZO$)
30 PRINT X

RUN
 13
```

Un stringa non può essere trattata direttamente come un valore.

```
?TYPE MISMATCH ERROR IN 20
READY.
```

Messa a punto dei programmi: RUN/STOP - STOP - CONT

I programmi non funzionano sempre "al primo colpo". Il BASIC invia messaggi per alcuni errori (per esempio di sintassi) ma non individua gli errori logici. Per i casi più delicati, occorre seguire passo per passo lo svolgimento del programma, cosa che, in BASIC, è relativamente semplice.

• Premendo STOP/RUN od aggiungendo un'istruzione STOP, possiamo interrompere l'esecuzione del programma.

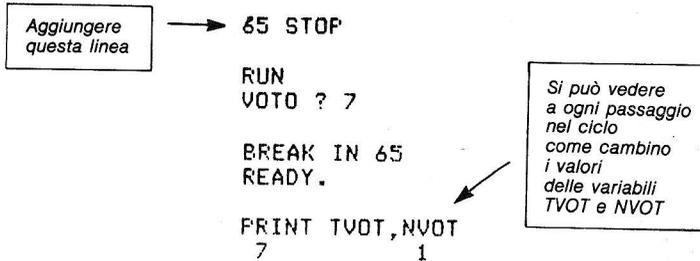
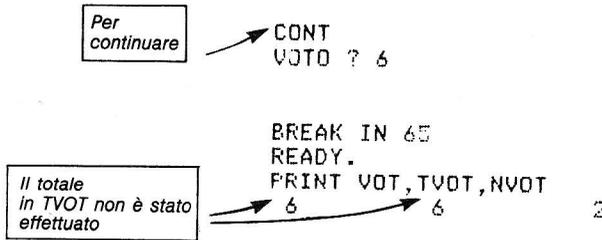
Possiamo allora visualizzare i valori delle variabili in modo diretto.

L'esecuzione interrotta può essere proseguita battendo CONT (continua).

Esempio

Il programma seguente effettua la media di molti voti.
Abbiamo commesso (volontariamente!) un errore: alla riga 50, invece di TVOT = TVOT + VOT, abbiamo scritto TVOT = VOT.

Naturalmente, la media ottenuta (6) è falsa.

• Sul Commodore 64 non possiamo bloccare un programma in attesa su di un'istruzione "INPUT" (con RUN /STOP).
Di conseguenza, per l'esempio in questione, dovremo utilizzare l'istruzione "STOP":

Per ora, niente di anormale.
Battiamo CONT per proseguire con l'esecuzione del programma ed interrompiamo di nuovo dopo avere immesso il secondo dato:

Ci accorgiamo, osservando il valore di TVOT, che il totale dei dati non è stato effettuato.

• Potremmo anche inserire delle righe per visualizzare i valori delle variabili:

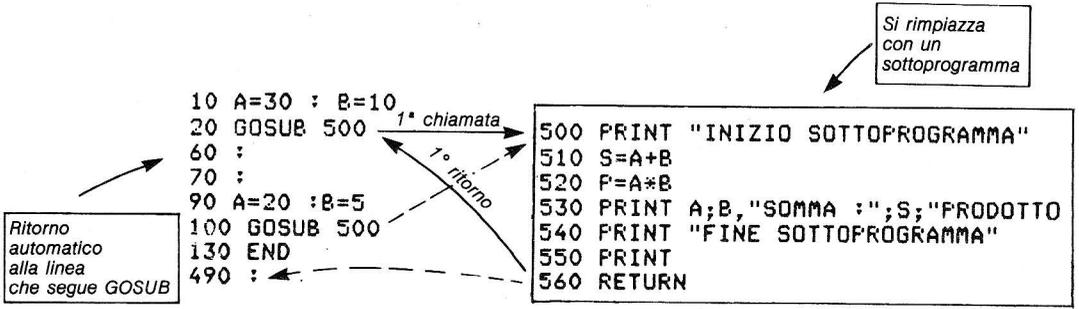
```
45 PRINT TVOT,VOT
55 PRINT TVOT,NVOT
```

• Quando l'errore è stato trovato e corretto, eliminiamo queste righe.

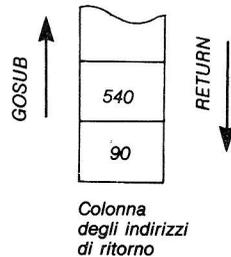
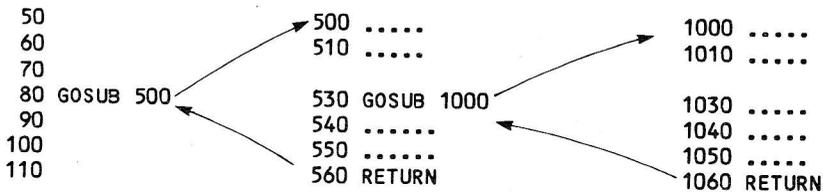
I sottoprogrammi: GOSUB/RETURN

Capita spesso che una stessa sequenza di istruzioni sia utilizzata PIÙ VOLTE in un programma.

Un sottoprogramma permette di scrivere UNA SOLA VOLTA questa sequenza, che verrà poi richiamata da diversi punti del programma per mezzo di GOSUB n. riga.

20 GOSUB 500 provoca un allacciamento del programma alla riga 500 (come lo avrebbe fatto GOTO 500), ma **l'istruzione RETURN (RITORNO) posta alla fine del sottoprogramma provoca un RITORNO AUTOMATICO all'istruzione che segue GOSUB 500,** cioè la riga 30 dell'esempio.
Per il secondo richiamo del sottoprogramma alla riga 100, il ritorno viene effettuato alla riga 130.

```
RUN
INIZIO SOTTOPROGRAMMA
 30   10     SOMMA : 40 PRODOTTO : 300
FINE SOTTOPROGRAMMA

INIZIO SOTTOPROGRAMMA
 20   5      SOMMA : 25 PRODOTTO : 100
FINE SOTTOPROGRAMMA
```

Nota: la riga 130 indica la fine del programma ed impedisce quindi (una volta eseguito il programma principale) di entrare nel sottoprogramma.

Un sottoprogramma può richiamarne a sua volta un altro.
Gli indirizzi di ritorno (91 e 540 nell'esempio) sono gestiti dal BASIC come se fossero incolonnati.

Le istruzioni sono eseguite nel seguente ordine:

Istruzioni	Stack	Descrizione
50,60,70	NIENTE	Programma principale
80,GOSUB 500	90	Richiamo 1^ sottoprogramma
500,510,520	90	Inizio 1^ sottoprogramma
530,GOSUB 500	540 90	Richiamo 2^ sottoprogramma
1000,1010,...,1050	540 90	1^ sottoprogramma
1060 RETURN	~~540~~ 90	Ritorno al 1^ sottoprogramma
540,550	90	1^ sottoprogramma
560 RETURN	~~90~~	Ritorno programma principale
90,100	NIENTE	Programma principale

Non cercate di entrare o di uscire da un sottoprogramma per mezzo di GOTO.

Interludio

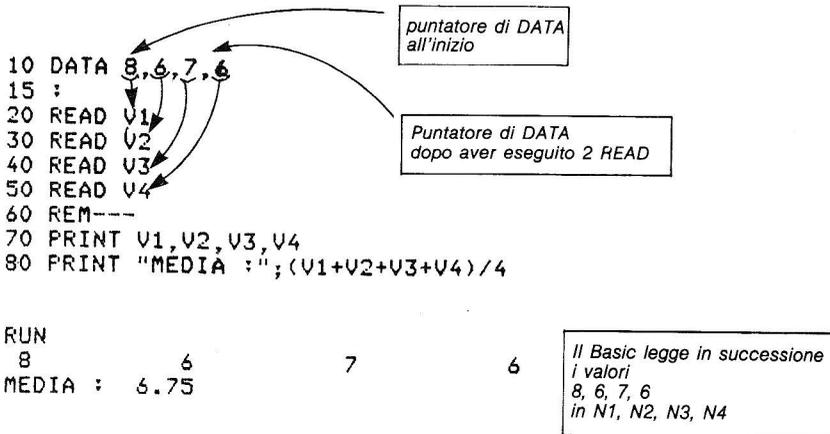
A proposito dei diagrammi di flusso

Bisogna utilizzare dei diagrammi di flusso?

Questo punto è molto controverso; molti sono favorevoli; altri, contrari, non li utilizzano.
I favorevoli ai diagrammi pretendono che un programma debba essere preceduto da un diagramma, che un programma da solo non può essere compreso direttamente.
Coloro che non li utilizzano affermano che un programma ben strutturato non abbia bisogno del diagramma, che un diagramma troppo sviluppato divenga ben presto confuso.

Noi pensiamo che **sia importante prestare molta cura alla presentazione dei programmi**, in modo tale che possano essere compresi con il minimo sforzo.
Un diagramma "segue" raramente un programma; rimane solamente il programma. **È per questo che un programma deve essere il più documentato possibile.**

DATA/READ/RESTORE

L'istruzione DATA permette di definire dei dati nel programma stesso. Essi vengono in seguito letti entro variabili per mezzo dell'istruzione "READ variabile" (LEGGERE variabile).

"READ N1" legge il primo dato (10) in V1.
Il puntatore di DATA (comandato dal BASIC) aumenta di 1.
Così "READ V2" legge il secondo dato in V2, ecc.
Nell'esempio seguente abbiamo supposto che il numero dati da leggere (4) fosse conosciuto.
Per trovare la fine dei dati mettiamo 99.

```
10 DATA 8,6,7,6,99
15 :
30 READ VOT
40 IF VOT=99 THEN GOTO 100
45 PRINT VOT,
50 TVOT=TVOT+VOT
60 NVOT=NVOT+1
70 GOTO 30
80 REM----------
100 PRINT "MEDIA :";TVOT/NVOT
```

Per indicare la fine del voto

```
RUN
 8           6           7           6
MEDIA :   6.75
```

I dati possono essere scritti su più righe:

```
10 DATA 8,6,7
20 DATA 6,7,99
```

è equivalente alla riga del programma seguente.
Le istruzioni DATA possono essere scritte all'inizio od alla fine del programma.

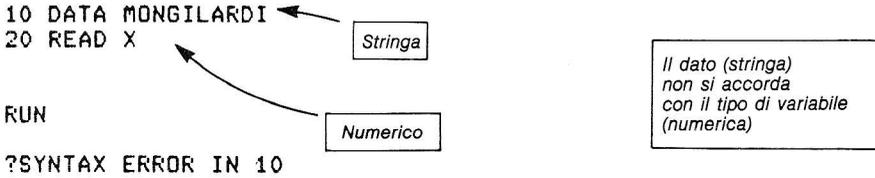
Caratteri speciali

Le istruzioni DATA contenenti caratteri speciali devono essere poste tra virgolette. Senza le virgolette, la virgola sarà considerata come separatrice.

```
10 DATA "VIA MERCANTI,15"
20 :
30 READ X$
40 PRINT X$

RUN
VIA MERCANTI,15
```

Errori

OUT OF DATA. Se il numero delle READ eseguite è superiore al numero dei dati presenti nell'istruzione DATA, appare il messaggio OUT OF DATA.

SYNTAX ERROR: il tipo di dato letto deve accordarsi con il tipo di variabile.

Occorre scrivere: 20 READ X$.

RESTORE

Si riposiziona all'inizio dell'istruzione DATA, permettendo così di rileggerla dal primo dato.

```
10 DATA 6,3,14
20 :
30 READ A
40 READ B
50 READ C
60 :
70 RESTORE
80 :
90 READ D,E,F
100 :
110 PRINT A,B,C
120 PRINT D,E,F


RUN
 6          3           14
 6          3           14
```

```
READY.
1 REM **** INIZIALIZZAZIONE BASIC ****
5 CONTATORE=0 : REM * CONTATORE DEI LOOP
10 REM * CARICAMENTO TABELLA(MESI)
15 DIM M$(12) : REM DICHIARAZIONE TABELLA
20 CONTATORE=CONTATORE+1
35 READ A$ : REM LETTURA DEI MESI IN DATA
40 M$(CO)=A$ : REM CARICAMENTO TABELLA
45 IF CONTATORE<12 THEN GOTO 20
50 CONTATORE=0 : REM INIZIALIZZAZIONE CONTATORE
110 PRINT "LA VOSTRA ETA'"; INPUT ETA'
115 PRINT : INPUT "MESE IN CIFRE";MESE
116 REM *** TEST ERRORI ***
117 IF ME<1 OR ME>12 THEN GOTO 115
120 PRINT : PRINT "MESE DI NASCITA";M$(ME)
130 NA=1984-ETA' : PRINT : PRINT "ANNO DI NASCITA";NA
140 PRINT : PRINT
```

(segue)

(*seguito*)

```
150 CONTATORE=CONTATORE+1 : IF CONTATORE<3 THEN GOTO 110
170 PRINT "FINE DEL PROGRAMMA"
200 DATA GENNAIO,FEBBRAIO,MARZO,APRILE
210 DATA MAGGIO,GIUGNO,LUGLIO,AGOSTO
220 DATA SETTEMBRE,OTTOBRE,NOVEMBRE,DICEMBRE


LA VOSTRA ETA' ? 17

MESE IN CIFRE ? 12

MESE DI NASCITA DICEMBRE

ANNO DI NASCITA 1967

LA VOSTRA ETA' ? 23

MESE IN CIFRE ? 16

MESE IN CIFRE ? 12

MESE DI NASCITA DICEMBRE

ANNO DI NASCITA 1961

FINE DEL PROGRAMMA
```

In breve

- *L'istruzione* ***DATA*** *permette di definire dei dati all'interno del programma. Questi dati sono separati da una virgola.*

- *Le DATA sono lette entro variabili per mezzo dell'istruzione* ***READ variabile.***

- *L'istruzione* ***RESTORE*** *si riposiziona all'inizio di DATA, il che permette di rileggerla dall'inizio. RESTORE non ha effetto sulle variabili.*

Le tabelle

Per memorizzare 4 voti, potremmo utilizzare 4 variabili V1, V2, V3, V4, facendo:

```
10 INPUT "VOTO 1";V1
20 INPUT "VOTO 2";V2
30 INPUT "VOTO 3";V3
40 INPUT "VOTO 4";V4
```

Utilizziamo invece una tabella che chiamiamo VOT( ).
Gli elementi di questa tabella sono identificati come VOT(1)
VOT(2)
VOT(3)
VOT(4)

Potremmo operare come nel seguente esempio:

```
10 INPUT "VOTO 1";VOT(1)
20 INPUT "VOTO 2";VOT(2)
30 INPUT "VOTO 3";VOT(3)
40 INPUT "VOTO 4";VOT(4)
```

Ma utilizziamo invece un ciclo FOR; facendo variare un "indice", semplificheremo la stesura del programma:

```
         10 FOR N=1 TO 4
4 volte  20 : INPUT "VOTO";VOT(N)
         30 NEXT N
```

LA 1ª volta è uguale a VOT(1) poiché N = 1

All'inizio, N è uguale ad 1. Di conseguenza

```
20 INPUT "VOTO";VOT(N) E' UGUALE A
20 INPUT "VOTO";VOT(1)
```

È dunque in VOT(1) che si introduce il primo voto.

```
8 ==> !  8! VOT(1)
      !  0! VOT(2)
      !  0! VOT(3)
      !  0! VOT(4)
      -----
```

Al secondo passaggio nel ciclo FOR, VOT(N) è uguale a VOT(2), dato che N è diventato uguale a 2.

Senza il ciclo FOR, avremmo dovuto fare:

```
10 N=1
20 : 3
30 INPUT "VOTO";VOT(N)
40 N=N+1
50 IF N=4 THEN 100
60 GOTO 30
70 :
100 STOP
```

Stampa della tabella VOT( )

Stampa della tabella VOT( ) dopo averla immessa.

```
10 FOR N=1 TO 4
20 INPUT "VOTO";VOT(N)
30 NEXT N
40 REM------------STAMPA DELLA TABELLA VOT()
50 PRINT
60 PRINT "LISTA DEI VOTI"
70 PRINT
80 FOR N=1 TO 4
90 :PRINT "VOTO";V;VOT(N)
100 NEXT N


RUN
VOTO ? 8
VOTO ? 6
VOTO ? 7
VOTO ? 6

LISTA DEI VOTI

VOTO 1  8
VOTO 2  6
VOTO 3  7
VOTO 4  6
```

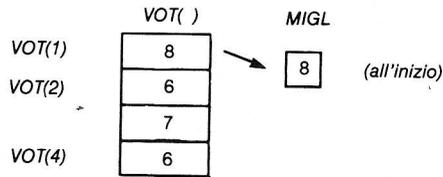
Dimensionamento delle tabelle: DIM (numero di elementi)

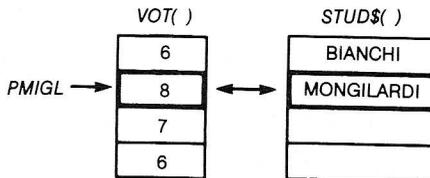
Le tabelle con oltre 10 elementi devono essere dimensionate, per riservare loro lo spazio in memoria centrale.
I voti potranno essere definiti in un'istruzione DATA:

```
10 DATA 8,6,7,6
20 :
25 DIM VOT(4)
30 FOR N=1 TO 4
40 :READ VOT(N)
50 NEXT N
60 REM-----------------TOTALE E MEDIA
70 TVOT=0
80 FOR N=1 TO 4
100 :TVOT=TVOT+VOT(N)
110 NEXT N
120 :
130 PRINT "TOTALE/MEDIA:";TVOT,TVOT/4

RUN
TOTALE/MEDIA: 27     7
```

Ricerca del miglior voto

Supponiamo da principio che il primo voto sia il migliore. Poi lo confrontiamo col secondo.

Se questo è maggiore, diventa il migliore, ecc.

```
10 DIM VOT(4)
20 REM----------INSERIMENTO VOTI
30 FOR N=1 TO 4
40 :INPUT "VOTO";VOT(N)
50 NEXT N
60 REM----------RICERCA DEL VOTO MIGLIORE
70 MIGL=VOT(1)
80 FOR N=2 TO 4
90 :IF VOT(N)>MIGL THEN MIGL=VOT(N)
100 NEXT N
110 REM---------
120 PRINT "IL VOTO MIGLIORE E' :";MIGL

RUN

IL VOTO MIGLIORE E': 8
```

Nell'esempio seguente, non memorizziamo il voto migliore ma la sua POSIZIONE nella tabella.

```
10 DIM VOT(4)
20 DIM STUD$(4)
30 REM----------INSERIMENTO VOTI
40 FOR N=1 TO 4
50 :INPUT "VOTO";VOT(N)
60 :INPUT "STUDENTE";STUD$(N)
70 NEXT N
80 REM----------
90 PMIGL=1
100 FOR N=2 TO 4
110 :IF VOT(N)>VOT(PMIGL) THEN PMIGL=N
120 NEXT N
130 REM---------
140 PRINT "MIGLIOR VOTO :";VOT(PMIGL),STUD$(PMIGL)
```

```
RUN
VOTO ? 6
STUDENTE ? BIANCHI
VOTO ? 8
STUDENTE ? MONGILARDI
VOTO ? 7
STUDENTE ? RASTELLI
VOTO ? 6
STUDENTE ? LODI
MIGLIOR VOTO:  8   MONGILARDI

PRINT PMIGL                  <---- Modo diretto
2
```

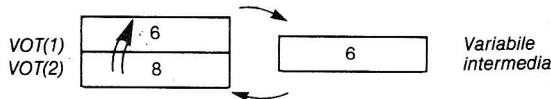
Nota: Le operazioni che abbiamo compiuto (totale, media, ricerca del miglior voto) non hanno bisogno di tabelle. Ma delle operazioni più complesse, quale ad esempio la selezione, necessitano l'utilizzo di tabelle.

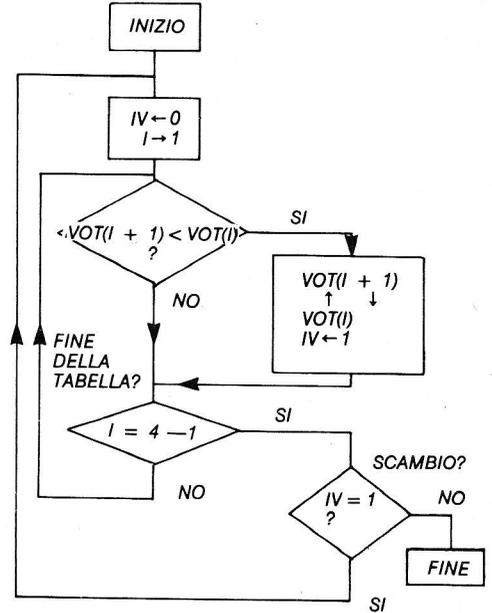
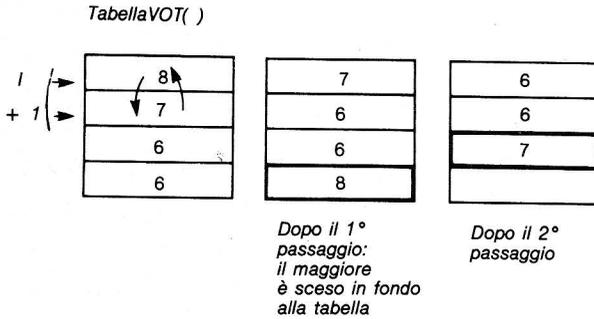
Selezione dei voti

Selezioniamo i voti contenuti nella tabella VOT( ).
Confrontiamo da principio il 2° elemento della tabella con il 1°:

- Se è maggiore, li lasciamo nell'ordine in cui sono.
- Se è minore, li invertiamo.

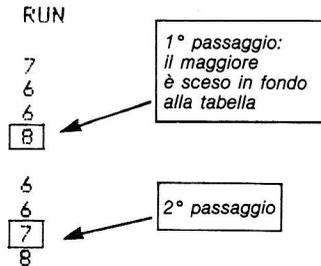
```
230 IF VOT(I+1) < VOT(I) THEN X=VOT(I) : VOT(I)=VOT(I+1) : VOT(I+1)=X : IV=1

     !                        !

   SE VOT(I+1) < VOT(I) ALLORA SCAMBIO DI VOT(I+1) E DI VOT(I)
```

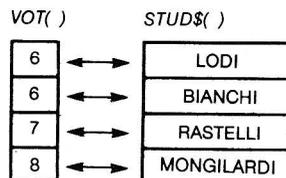
L'inversione di VOT(I) e di VOT(I+1) si compie utilizzando una variabile intermedia (X):

In seguito, confrontiamo con lo stesso metodo il terzo elemento con il secondo (incrementando l'indice di 1) e li invertiamo se non sono nel giusto ordine, ecc.
Dopo aver confrontato tutti gli elementi della tabella, il maggiore di essi è sceso in fondo alla tabella. (Ricordate che ci sono N-1 comparazioni per N elementi). Ma la selezione non è necessariamente completata.
Per saperlo, basta osservare il campo di controllo IV. Se non si verificano inversioni durante l'esplorazione della tabella, dovrà essere VOT(1) $<$ VOT(2) $<$ VOT(3) $<$ VOT(4). Di conseguenza, gli elementi sono già in ordine.
Se invece si verificano inversioni, esploriamo di nuovo la tabella. Al termine della

seconda selezione, il maggiore degli N-1 elementi è arrivato in penultima posizione. Sono necessarie al massimo N esplorazioni per completare la selezione della tabella.

```
40 FOR I=1 TO 4
50 :INPUT "VOTO";VOT(I)
60 NEXT I
200 REM----------SELEZIONE DEI VOTI
205 :
210 IV=0
220 FOR I=1 TO 4-1
230 :IF VOT(I+1)<VOT(I) THEN X=VOT(I):VOT(I)=VOT(I+1):VOT(I+1)=X:IV=1
240 NEXT I
250 IF IV=1 THEN 210
255 :
260 REM----------STAMPA DEI VOTI SELEZIONATI
270 FOR I=1 TO 4
280 :PRINT VOT(I)
290 NEXT I

RUN
VOTO ? 8
VOTO ? 7
VOTO ? 6
VOTO ? 6
 6
 6
 7
 8
```

Per seguire più dettagliatamente l'evoluzione della selezione potete inserire:

```
245 FOR K=1 TO 4:PRINT VOT(K):NEXT K:PRINT
```

Dopo ogni selezione, stampiamo il contenuto della tabella.

Per ottenere la lista dei voti in ordine crescente accompagnati dalla lista dei nomi, basta far compiere un'inversione nella tabella STUD$( ) ogni volta che ne compiamo una in VOT( ).

```
40 FOR I=1 TO 4
50 :INPUT "VOTO";VOT(I)
55 :INPUT "NOME DELLO STUDENTE";STUD$(I)
60 NEXT I
200 REM-----------SELEZIONE DEI VOTI
205 :
210 IV=0
220 FOR I=1 TO 4-1
230 :IF VOT(I+1)=>VOT(I) THEN 240
235 :X=VOT(I):VOT(I)=VOT(I+1):VOT(I+1)=X
237 :X$=STUD$(I):STUD$(I)=STUD$(I+1):STUD$(I+1)=X$:IV=1
240 NEXT I
250 IF IV=1 THEN 210
255 :
260 REM-----------STAMPA DEI VOTI SELEZIONATI
270 FOR I=1 TO 4
280 :PRINT VOT(I),STUD$(I)
290 NEXT I
```

Per ottenere la lista selezionata in ordine crescente dei nomi, fate:

```
230 IF STUD$(I+1)=>STUD$(I) THEN 240
```

Sistemazione di una fattura

Nelle DATA sono definiti dei prodotti (referenza, definizione e prezzo).
I prodotti fatturati sono collocati in 4 tabelle RIF$( ), DEF$( ), PREZZO( ) e QT( ).
Quando tutti i prodotti saranno stati inseriti, stamperemo la fattura.

Esecuzione

```
NOME CLIENTE ? ROTONDI
INDIRIZZO ? VIA XXXXXXXXXXX 36

PRODOTTO (F PER FINE) ? PR 1
QUANTITA' ?             3
PRODOTTO (F PER FINE) ? PR 2
QUANTITA' ?             2


STABILIMENTO XXXXXX
                   FATTURA DI : ROTONDI XXXXXX
                                VIA XXXXXXXXXXX 36

PR 1    TAVOLO XXX        200000   3       600000
PR 3    SEDIA XXXX         80000   2       160000


                              TOTALE :   760000
```

Fattura

```
100 REM FT              FATTURA
110 :
120 REM------------RIFERIMENTI,DEFINIZIONI E PREZZO DEI PRODOTTI
130 :
140 DATA "PR 1","POLTRONA XXXX",125000
150 DATA "PR 2","TAVOLO XXCXXXXXX",200000
160 DATA "PR 3","SEDIA XXXXXX",80000
170 DATA *
180 REM------------SCELTA DEI RIFERIMENTI
190 INPUT "NOME CLIENTE";NOME$
200 INPUT "INDIRIZZO";IND$
210 :
220 NL=0                :REM NL=N● RIGHE DI FATTURA
230 :
240 INPUT "RIFERIMENTI PRODOTTI (F PER FINE)";R$
250 IF R$="F" THEN 420
260 INPUT "QUANTITA'";QT
270 :
280 RESTORE
290 :
300 READ RIF$:IF RIF$=* THEN PRINT "NON ESISTE":GOTO 240
310 READ DEF$
320 READ PREZZO
330 IF R$< >RIF$ THEN 300
340 NL=NL+1
350 RIF$(NL)=RIF$
360 DEF$(NL)=DEF$
370 PREZZO(NL)=PREZZO
380 QT(NL)=QT
390 :
400 GOTO 240
410 REM------------STAMPA FATTURA
420 PRINT
430 PRINT "STABILIMENTO XXX":PRINT
440 PRINT TAB(15);"FATTURA DI :";NOME$
450 PRINT TAB(15);IND$
460 PRINT
470 T=0
480 FOR L=1 TO NL
490 :PRINT RIF$(L)
500 :PRINT TAB(5);DEF$(L)
510 :PRINT TAB(23);PREZZO(L)
520 :PRINT TAB(29);QT(L)
530 :PRINT TAB(33);QT(L)*PREZZO(L)
540 :T=T+QT(L)*PREZZO(L)
550 NEXT L
560 PRINT
570 PRINT TAB(25);"TOTALE :";TAB(33);T
```

Scegliete il vostro menu

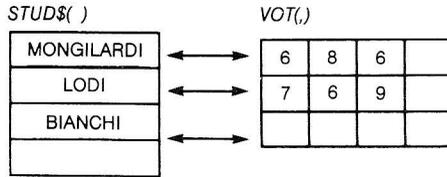
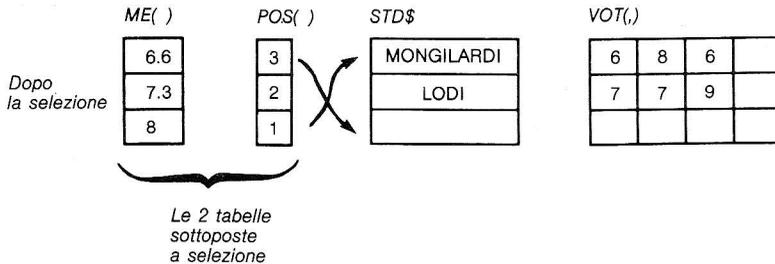
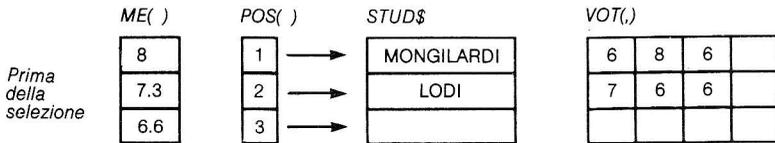
Potete scegliere tra 3 antipasti, 3 primi e 3 contorni.
Disponete solamente di 12.000 lire.
Sono calcolate tutte le combinazioni di antipasti, di primi e di contorni, occorre definirli in DATA e leggerli in 3 tabelle (come per i prezzi).

```
100 REM      RISTORANTE
110 :
120 DEN=#12000                  :REM DENARO DISPONIBILE
130 REM------------------PREZZO DI ANTIPASTI,PRIMI PIATTI E CONTORNI
140 DATA 1600,2400,2000
150 DATA 7000,8000,9000
160 DATA 2400,3000,4000
170 :
180 FOR A=1 TO 3 : READ PA(A):NEXT A
190 FOR P=1 TO 3 : READ PP(P):NEXT P
200 FOR C=1 TO 3 : READ PC(C):NEXT C
210 :
220 FOR A=1 TO 3
230 :FOR P=1 TO 3
240 : FOR C=1 TO 3
250 :  T=PA(A)+PP(P)+PC(C)
260 :  IF T>DEN THEN 260
270 :  PRINT "ANTIPASTI :";A;"PRIMI :";P;"CONTORNI :";C;"TOTALE :";T
280 : NEXT C
290 :NEXT P
300 NEXT A
```

PA( )	*PP( )*	*PC( )*
1600	7000	2400
2000	8000	9000
2400	3000	4000
Prezzo antipasti	*Prezzo primi*	*Prezzo contorni*

```
RUN
ANTIPASTI :1   PRIMI :1   CONTORNI :1     TOTALE :  11000
ANTIPASTI :1   PRIMI :1   CONTORNI :2     TOTALE :  11600
ANTIPASTI :1   PRIMI :2   CONTORNI :1     TOTALE :  12000
ANTIPASTI :2   PRIMI :1   CONTORNI :1     TOTALE :  11800
ANTIPASTI :3   PRIMI :1   CONTORNI :1     TOTALE :  11400
ANTIPASTI :3   PRIMI :1   CONTORNI :2     TOTALE :  12000
```

Tabelle a due dimensioni

Le tabelle possono essere a più dimensioni (fino a 255).
Abbiamo un insieme di campi definiti in DATA e li vogliamo stampare. Potremmo leggere le DATA e stamparli via via (vedere il capitolo sulle DATA).
L'uso di una tabella a 2 dimensioni permette di effettuare altre operazioni (come ad esempio la selezione). Ogni riga rappresenta il voto di uno studente.

Per ottenere la lista degli studenti e dei loro voti nell'ordine decrescente delle medie, utilizzeremo una tabella intermedia (PT).
Questa tabella PT( ) "punta" alla tabella dei voti VOT(,). La selezione è fatta sulla tabella ME( ) invertendo gli elementi di PT( ) ogni volta che invertiamo 2 elementi di ME( ).
Senza la tabella PT( ), si dovrebbero invertire i voti della tabella VOT(,), il che sarebbe molto scomodo.

Analisi dei voti

```
100 REM ANVOT                 ANALISI DI VOTI
110 :
120 DATA MONGILARDI,  6,8,6,99
130 DATA LODI,  7,6,9,99
140 DATA BIANCHI,  10,4,10,99
150 DATA *
160 REM-------------------------LETTURA DATA
170 DIM STUD$(30),ME(30),POS(30)
180 DIM VOT(30,10)
190 :
200 FOR E=1 TO 30
210 :READ X$:IF X$="*" THEN NE=E-1 :GOTO 300
220 :STUD$(E)=X$
230 :
240 :FOR N=1 TO 10
250 :READ X: IF X=99 THEN 280
260 :VOT(E,N)=X
270 :NEXT N
280 NEXT E
290 REM-------------------------STAMPA DI STUD$() E DI VOT(,)
300 FOR E=1 TO NE
310 :PRINT STUD$(E);
320 :FOR N=1 TO 10
707 :IN VOT(E,N)=0 THEN 360
340 :PRINT TAB(12+N*3);VOT(E,N);
250 NEXT N
360 :PRINT
370 NEXT E
380 REM--------------CALCOLO DELLE MEDIE IN ME()---
390 PRINT:PRINT"MEDIE:":PRINT
400 FOR E=1 TO NE
410 :TN=0
420 :FOR N=1 TO 10
430 : IF VOT(E,N)=0 THEN NV=N-1:GOTO 460
440 : TV=TV+VOT(E,N)
450 :NEXT N
460 :ME(E)=TV/NV
470 :PRINT "MED:";STUD$(E);TAB(20);ME(E)
480 NEXT E
490 REM=========================STAMPA NELL' ORDINE DELLE MEDIE
500 FOR I=1 TO 30 :POS(I)=I:NEXT I
510 REM----TEST
520 IV=0
530 FOR I=1 TO NE-1
540 :IF ME(I+1)<=ME(I) THEN 570
550 :X=ME(I):ME(I)=ME(I+1):ME(I+1)=X
560 :X=POS(I):POS(I)=POS(I+1):POS(I+1)=X:IV=0
570 NEXT I
580 IF IV=1 THEN 520
590 REM------------------STAMPA
600 PRINT:PRINT "LISTA NELL' ORDINE DELLE MEDIE":PRINT
610 FOR E=1 TO NE
620 :PRINT STUD$(POS(E));
630 :FOR N=1 TO 10
640 : IF VOT(POS(E),N)=0 THEN 660
```

(segue)

(seguito)

```
650 : PRINT TAB(10+3*N);VOT(POS(E),N);
660 :NEXT N
670 :PRINT TAB(26);ME(E)
680 NEXT E

READY.


MONGILARDI           6    8    6
LODI                 7    6    9
BIANCHI             10    4   10

MEDIE :

MED: MONGILARDI      6.6666
MED: LODI            7.3333
MED: BIANCHI         8

LISTA NELL' ORDINE DELLE MEDIE

BIANCHI             10    8   10    8
LODI                 7    6    9    7.3333
MONGILARDI           6    8    6    6.6666
```

Elezioni

Vengono gestiti numerosi seggi. I risultati per ogni candidato ed ogni seggio vengono collocati in una tabella VX(,) a 2 dimensioni.

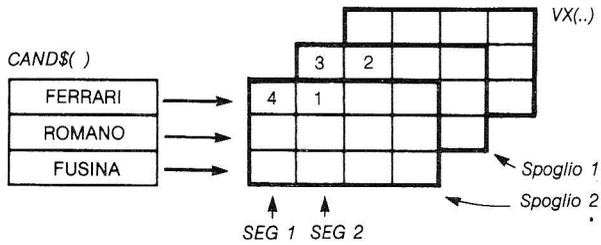
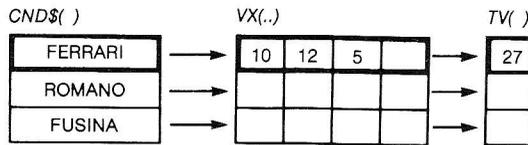
Un primo modo (CAND) permette di definire i nomi dei candidati e di visualizzare la tabella CND$( ).

```
SCELTA ? CAND
NOME CANDIDATO ? (F PER FINE)        FERRARI
NOME CANDIDATO ? (F PER FINE)        FUSINA
NOME CANDIDATO ? (F PER FINE)        ROMANO
NOME CANDIDATO ? (F PER FINE)        F
```

Un secondo modo (VSEG) permette di visualizzare la tabella VX(,) per un seggio (1,2,...). Nell'esempio, non viene effettuato il totale parziale durante lo spoglio; bisogna inserire il numero totale dei voti per un seggio. Il vecchio valore può comunque essere modificato.

Un terzo modo visualizza i risultati (RISUL): totale dei voti per ogni candidato e percentuale.
Per effettuare il totale per ogni seggio durante lo spoglio, vi sono 2 soluzioni:

- Con la stessa tabella a 2 dimensioni VX(,), aggiungere via via i voti.

- Definire la tabella VX con 3 dimensioni e collocarvi tutti gli spogli parziali. Questa soluzione è più "sicura" poiché permette di rintracciare ogni spoglio.

Per ottenere la lista dei risultati nell'ordine dei voti ottenuti da ciascun candidato, potremmo creare una tabella TV( ) e selezionarla.

```
A/ Totale nella tabella TV()

600 FOR C=1 TO NC
610 :TV(C)=0
620 :FOR B=1 TO 5
630 :  TV(C)=TV(C)+VX(C,B)
640 :NEXT B
650 NEXT C

B/ Test di TV() e CND$()

700 IV=0
710 FOR I=1 TO NC-1
715 :IF TV(I+1)<=TV(I) THEN 730
720 :X=TV(I):TV(I)=TV(I+1):TV(I+1)=X
725:X$=CND$(I):CND$(I)=CND$(I+1):CND$(I+1)=X$
730 NEXT I
740 IF IV=1 THEN 700

C/ Stampa di TV() e CND$()

800 FOR C=1 TO NC
810 : PRINT TV(C),CND$(C)
820 NEXT C


60 FERRARI
44 FUSINA
27 ROMANO
```

Elezioni

```
10 REM EL     ELEZIONI
20 :
50 DIM CND$(10),VX(10,5)          :REM 10 CANDIDATI/5 SEGGI
60 REM--------------------SCELTA------------------
75 PRINT CHR$(147)
80 PRINT:INPUT "VOSTRA SCELTA(CAND,SEG,RISUL)";SC$
90 IF SC="CAND" THEN GOSUB 140
100 IF SC="SEG" THEN GOSUB 240
110 IF SC="RISUL" THEN GOSUB 350
120 GOTO 80
130 REM-------------------NOMI DEI CANDIDATI (CAND)--------
140 PRINT :C$=" ":INPUT "NOME CANDIDATO (F PER FINE)";C$
150 IF C$="F" OR C$=" " THEN 220
160 FOR C=1 TO 10
170 :IF C$=CND$(C) THEN PRINT:PRINT "E' GIA' PRESENTE":GOTO 140
180 :IF CND$(C)=" " THEN CND$(C)=C$: NC=C: GOTO 140
190 NEXT C
200 PRINT "TROPPI CANDIDATI":STOP
220 RETURN
230 REM-------------------SEGGI-------------------
240 PRINT:B=0:INPUT "QUALE SEGGIO (O PER FINE)";B:IF B=0 THEN RETURN
250 :
260 FOR C=1 TO NC
270 : PRINT CND$(C);TAB(10);
280 :PRINT VX(C,B);TAB(15);
290 :V=0:INPUT"VOTO (NUMERO O 0);V:IF V=0 310
300 :VX(C,B)=V
310 NEXT C
320 GOTO 240
330 REM--------------------VISUALIZZAZIONE RISULTATI-----
350 T=0
360 FOR C=1 TO NC
370 : FOR B=1 TO 5:T=T+VX(C,B):NEXT B
380 NEXT C
390 IF T=0 THEN PRINT "NESSUN VOTO ":RETURN
400 PRINT:PRINT"RISULTATI":PRINT
410 FOR C=1 TO NC
420 :PRINT CND$(C);
430 :TC=0
440 :FOR B=1 TO 5
450 :   TC=TC+VX(C,B)
460 :   IF VX(C,B)<>0 THEN PRINT TAB(6+B*4);VX(C,B);
470 :NEXT B
480 :PRINT TAB(26);TC;
490 :PC=TC/T*100: PRINT TAB(31);"%";INT(PC*100+.5)/100
500 NEXT C
510 RETURN
```

```
RISULTATI

FERRARI      20     30     20       70   % 49.65
ROMANO        8     12      7       27   % 19.15
FUSINA       20     14     10       44   % 31.21
```

ON A GOTO n. riga 1, n. riga 2, ...
ON A GOSUB n. riga 1, n. riga 2, ...

Collegamento multidirezionale

Secondo il valore di una variabile A (o di una espressione) 1, 2, 3,... ci si allaccia alla riga 1, riga 2, riga 3,...

Esempio:

Programmate una videata, ed assegnate ad alcuni caratteri una funzione ben definita.

H	Traccia un tratto orizzontale verso destra.
SHIFT/H	Traccia un tratto orizzontale verso sinistra.
V	Traccia un tratto verticale verso il basso.
SHIFT/V	Traccia un tratto verticale verso l'alto.

Soluzione con la condizione (IF... THEN)

```
10 GET A$:IF a$=" " GOTO 10 : REM   ATTESA CARATTERE DALLA TASTIERA
20 IF A$="h" GOTO 200 :REM  DESTRA
30 IF A$="H" GOTO 300 :REM  SINISTRA
40 IF A$="v" GOTO 400 :REM  DESTRA
50 IF A$="V" GOTO 500 :REM  SINISTRA
60 :
70 :
80 :
90 GOTO 10:REM NESSUNA CONDIZIONE
```

Soluzione con ON A GOTO n. riga 1, n. riga 2, ...

```
10 NB=4:DIM TB(NB):REM NUMERO DELLE CONDIZIONI
20 FOR I=1 TO NB
30 READ TB(I):NEXT :REM CARICA LA TABELLA
40 GET A$:IF A$=" " GOTO 140: REM ATTENDE UN CARATTERE DALLA TASTIERA
50 A=ASC(A$)
60 FOR I=1 TO NB:IF A<>TB(I) THEN NEXT:GOTO 140
70 A=I:I=NB:NEXT
80 ON A GOTO 200,300,400,500
90 DATA 72,200,86,214
```

10 Dichiara la tabella.
30 Carica i codici ASCII dei tasti nella tabella.
40 Aspetta che venga battuto un carattere da tastiera.
50 Trasforma il carattere in codice ASCII.
60 Cerca la sua posizione nella tabella. Se non gli corrisponde alcun codice ritorna alla 40.
70 "A" prende il valore di "I" (indice della posizione nella tabella) "I" prende il valore finale del ciclo e si chiude il loop.
80 Direzione secondo A.

La riga 90 contiene i codici ASCII dei tasti:

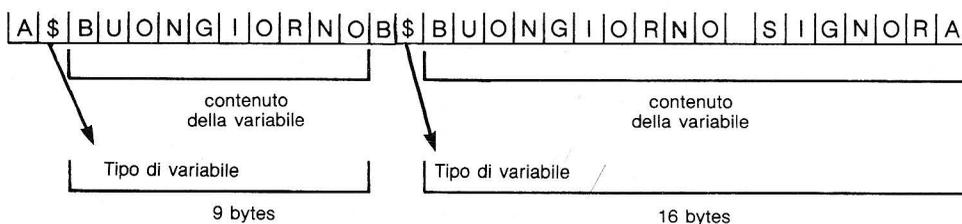
H = 72 — SHIFT/H = 200
V = 86 — SHIFT/V = 214
ecc.

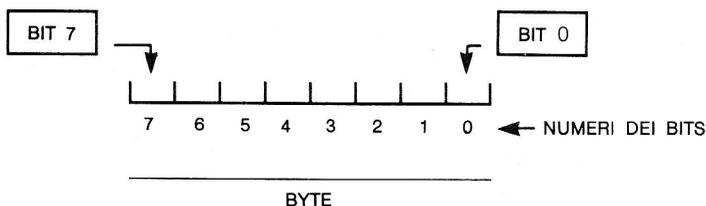
PEEK - POKE AND - OR

Tutte le informazioni vengono collocate in "caselle di memoria" chiamate bytes.

ATTENZIONE: non confondere "**casella di memoria**" con "**casella variabile**". In effetti una casella variabile è costituita da più caselle di memoria secondo l'importanza della variabile in questione.

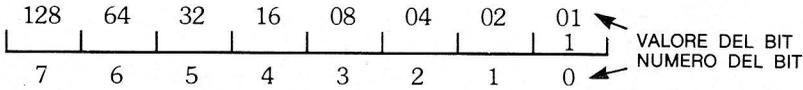
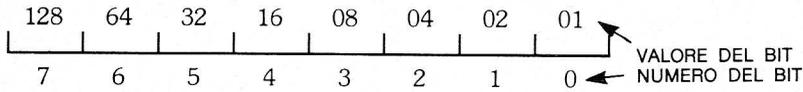
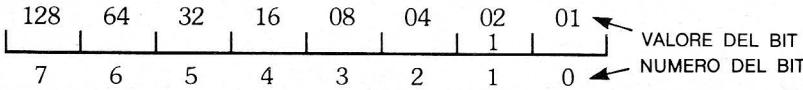
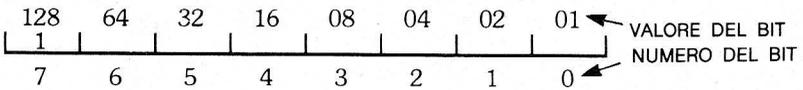
Esempio: A$ = "BUONGIORNO"
B$ = "BUONGIORNO SIGNORA"

Schematizziamo il modo in cui vengono collocate queste due variabili in memoria.

La variabile B$ occupa 7 bytes in più della variabile A$.

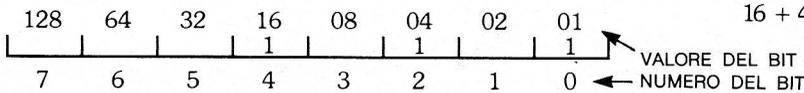
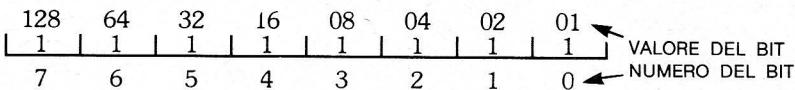
Ogni byte è diviso in 8 parti chiamate "Bit". Questi bits sono numerati, da destra a sinistra, da 0 a 7.

Un bit può avere solamente due stati: 0 o 1. Un bit a zero ha valore nullo. Un bit a 1 ha un valore differente a seconda della sua posizione nel byte.

128	64	32	16	08	04	02	01	← VALORE DEL BIT
7	6	5	4	3	2	1	0	← NUMERO DEL BIT

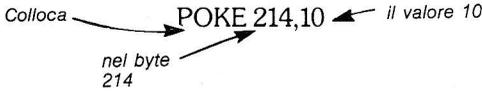
128	64	32	16	08	04	02	01	← VALORE DEL BIT	
							1		
7	6	5	4	3	2	1	0	← NUMERO DEL BIT	= 1 = 2 ↑ 0

128	64	32	16	08	04	02	01	← VALORE DEL BIT	
						1			
7	6	5	4	3	2	1	0	← NUMERO DEL BIT	= 2 = 2 ↑ 1

128	64	32	16	08	04	02	01	← VALORE DEL BIT	
1									
7	6	5	4	3	2	1	0	← NUMERO DEL BIT	= 128 = 2 ↑ 7

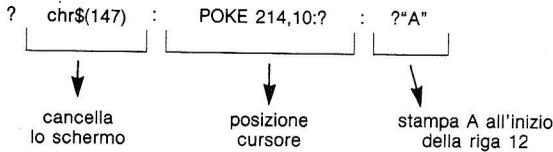
La freccia ↑, orientata verso l'alto, è il simbolo dell'elevazione a potenza.
Per codificare il valore 21 in un byte, occorre mettere a 1 i bits:

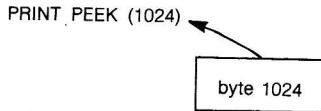
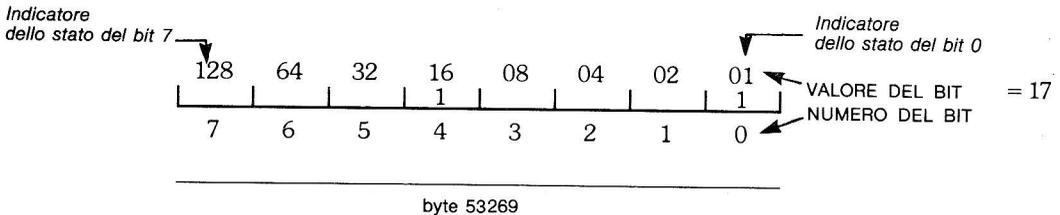
— 0 valore 1
— 2 valore 4
— 4 valore 16
TOTALE 21

Il valore massimo di un byte è 255.

POKE

L'istruzione POKE sostituisce il valore contenuto in un byte con un nuovo valore.

Esempio: si posizioni il cursore sulla linea 12 (10 + 2).

? chr$(147) : POKE 214,10:? : ?"A"

cancella lo schermo — posizione cursore — stampa A all'inizio della riga 12

PEEK

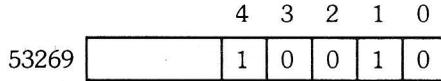
L'istruzione PEEK legge un byte senza modificarne il contenuto.

Il byte 1024 rappresenta la prima casella in alto a sinistra dello schermo.
Il valore derivato da "?PEEK(1024)" rappresenta il codice video del carattere che figura in quella casella (vedi appendice codici video).
Ad esempio, 32 rappresenta il codice spazio (blank).
Il byte 53269 definisce gli stati attivi relativi ai bits a video (confronta il capitolo valori relativi dei bytes).
A ciascun bit è associato uno stato (1 = attivo) (STATO 0 per Bit 0, STATO 1 per Bit 1, ecc.)

- Facendo POKE 53269,17 rendiamo attivi gli stati relativi ai bits 0 e 4.

- Se il byte 53269 contiene un valore nullo, possiamo commutare successivamente gli stati dei bits 0 e 4 facendo:

POKE 53269, PEEK (53269)+ 1 (VALORE BIT 0)
POKE 53269, PEEK (53269)+16 (VALORE BIT 4)

Il byte 53269 contiene ora 17.

• Ma se aggiungiamo nuovamente 1, il byte 53269 conterrà 18.

		4	3	2	1	0
53269		1	0	0	1	0

• È lo stato del bit 1 che viene attivato (invece di quello del bit 0).

• L'impiego dell'**addizione aritmetica** per portare lo stato di un bit a 1 ha senso solo se lo stato del bit è 0.

• Ora utilizzeremo l'operazione "**OR**" ("oppure" logico), che permette di portare dei bits a 1 anche se hanno già questo valore.

OR

L'istruzione **OR** effettua un "oppure" bit per bit fra 2 valori binari (espressi in decimali).

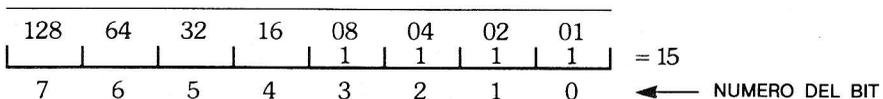
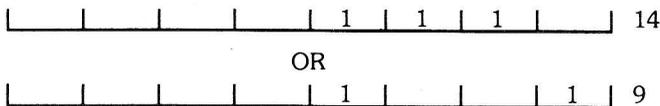
A = 14 OR 9.

Memorizza in A

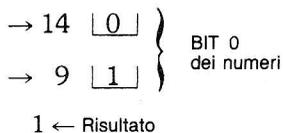
Il risultato è collocato in un byte o in una variabile.

REGOLA DELL'OPERAZIONE OR

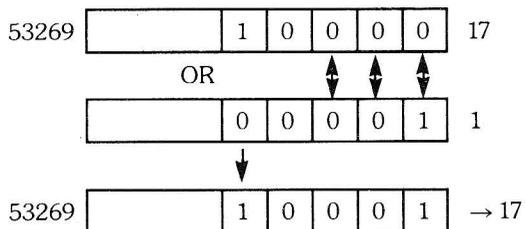
0	1	1	0
0	1	0	1
0 Risultato.	1 Risultato.	1 Risultato.	1 Risultato.

• Operazione "OR" fra 14 e 9.

Qui sotto osserviamo in dettaglio l'operazione per i due bits di destra.

→ 14 [0]
→ 9 [1]
BIT 0 dei numeri

1 ← Risultato

• Supponiamo che il byte contenga 17 (bits attivi 0 e 4) e che vogliamo attivare il bit 0 (che lo è già) facendo:

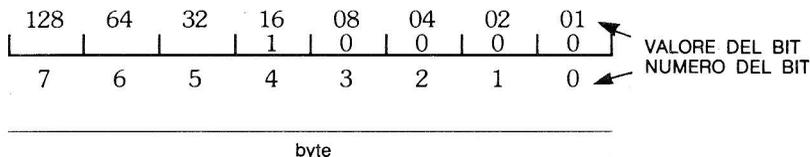
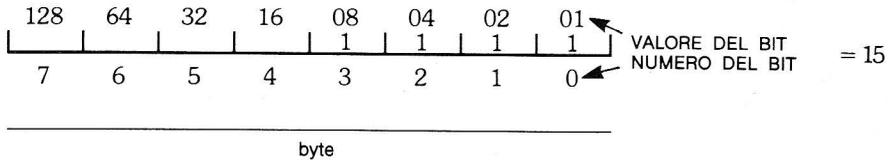
POKE 53269, PEEK (53269) OR 1

Non variamo il contenuto del byte 53269, che resta uguale a 17:

Ora disattiviamo il bit 0 sottraendo 1 al byte 53269:

POKE 53269, PEEK (53269) − 1

17 − 1 = 16

128	64	32	16	08	04	02	01	← VALORE DEL BIT
			1	0	0	0	0	
7	6	5	4	3	2	1	0	← NUMERO DEL BIT

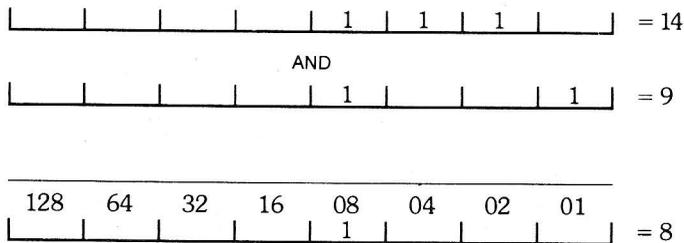
byte

Il bit 0 è disattivato.
Ma se sottraiamo ancora 1, non otteniamo il risultato cercato:

Per ottenere il risultato voluto, occorre utilizzare l'istruzione **AND**.

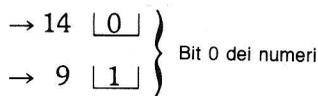
AND

L'istruzione **AND** effettua un "E" bit per bit fra due valori binari (espressi in decimali).

REGOLA DELL'OPERAZIONE AND

0	1	1	0
0	1	0	1
0 Risultato.	1 Risultato.	0 Risultato.	0 Risultato.

Operazione AND fra 14 e 9.

Qui sotto analizziamo in dettaglio l'operazione per i bits di destra:

→ 14 [0] }
→ 9 [1] } Bit 0 dei numeri

0 ← risultato

Battendo in modo diretto: PRINT 14 AND 9 ↵ otterrete 8.

• Per disattivare il bit 0 lasciando attivi tutti gli altri, effettueremo un AND fra il contenuto di 53269 e 254 (11111110 in forma binaria). Solamente il bit 0 è messo a 0.

POKE 53269, PEEK (53269) AND 254

• Effettuando di nuovo la stessa operazione, il risultato è identico.

In breve

• *L'istruzione* **POKE** *colloca un valore (0-255) in un byte.*

• *L'istuzione* **PEEK** *legge il valore di un byte.*

• *L'istruzione* **OR** *effettua un "OPPURE" bit per bit fra 2 valori binari.*

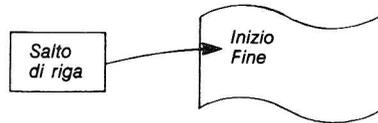
• *L'istruzione* **AND** *effettua un "E" bit per bit fra 2 valori binari.*

Le stampe

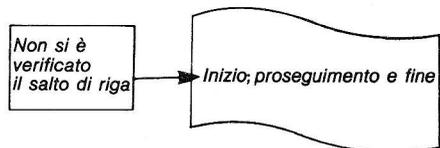
Presentiamo per sommi capi i metodi di stampa più usati.

PRINT variabile

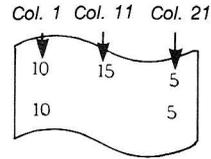
PRINT seguito da una variabile o da una costante ne provoca la stampa seguita da un salto di linea e da un ritorno a capo.
PRINT da solo provoca un salto di linea.

```
10 PRINT "INIZIO"
20 PRINT
30 PRINT "FINE"
```

PRINT variabile ;

Il ; (punto e virgola) impedisce il salto di linea.

```
10 PRINT "INIZIO";
20 PRINT "PROSEGUIMENTO E FINE"
```

PRINT variabile ,

La , (virgola) permette di stampare i valori inquadrati in modo standard nelle colonne 1, 11, 21,...

```
10 X=10:Y=15:Z=5
20 PRINT X,Y,Z
30 PRINT X, ,Z
```

PRINT TAB (posizione)

Posiziona il cursore alla colonna specificata dall'istruzione, permettendo così di inquadrare i dati.

```
10 INPUT "IL VOSTRO NOME";NOME$
20 :
30 FOR A=0 TO 6.28*3 STEP 6.28/20
40 X=20+SIN(A)*10
50 PRINT TAB(X);NOME$
60 NEXT A
```

```
              BIANCHI
                BIANCHI
                 BIANCHI
                BIANCHI
               BIANCHI
             BIANCHI
           BIANCHI
         BIANCHI
        BIANCHI
         BIANCHI
           BIANCHI
```

Disegno di un triangolo pieno

```
10 FOR L=1 TO 9 STEP 2
20 PRINT TAB(20-L/2);
30 FOR E=1 TO L
40 PRINT "*"
50 NEXT E
60 NEXT L

RUN
```

```
    *
   ***
  *****
 *******
*********
```

Sistemi di numerazione

Per rappresentare il numero 234, potremmo "disegnare" 234 aste. Ma è più semplice raffigurarlo come:

4 unità
3 decine (3 gruppi di 10 aste)
2 centinaia (2 gruppi di 100 aste)

10 rappresenta la "base". Potremmo anche utilizzare una base 8:

```
   !!!       !!!!!       !!
    3          5          2
    ↓          ↓          ↓
  3*64   +   5*8    +    2*1      -------> 234   in sistema decimale
```

In generale, un numero N in base B si esprime in questo modo:

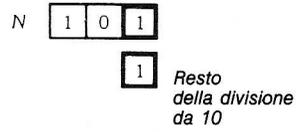
```
           n           n-1                       1         0
    N=an*B    + an-1*B       + ..... + a1*B   + a0*9
```

1101 in base 2 è uguale a:

```
    3        2        1        0
1*2   + 1*2   + 0*2   + 1*2        --------->1*8 + 1*4 + 0*2 + 1   ------->   13
```

```
10 REM CONVD         CONVERSIONE BASE B ----> DECIMALE
20 :
30 INPUT "BASE";B
40 :
50 INPUT "NUMERO";N
60 ND=0
70 :
80 FOR P=0 TO 6
90 :Q=INT(N/10)      :REM   QUOZIENTE
100 :R=N-(Q*10)      :REM   RESTO (DA' IL COEFFICIENTE DI GRADO P)
110 :
120 :ND=ND+(B P)*R
130 :N=Q
140 NEXT P
150 :
160 PRINT ND
170 GOTO 50
```

N | 1 | 0 | 1 |

| 1 | Resto della divisione da 10

```
RUN BASE ? 2
NUMERO ? 101
  5
```

Conversione decimale in base B

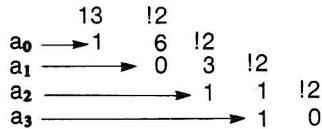
Esprimiamo un numero N sotto un'altra forma.

$$N = an * B^n + an-1 * B^{n-1} + ... + a0 * B^0 = \underbrace{(an * B) + an-1) * B + ... + a2) * B + a1)}_{Quoziente} * B + \underbrace{a0}_{Resto}$$

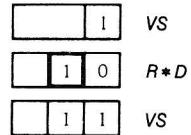
Dividendo N per B, otteniamo un quoziente Q = (an * B) + an — 1) * B +...+ a2) * B + a1) ed un resto uguale ad a0.
Dividendo il quoziente per B, otteniamo un resto uguale ad a1.
Così i resti di divisioni successive per la base B danno i coefficienti a0, a1,..., an.

Esempio

Convertiamo 13 (decimale) in base 2.

```
          13   !2
a0 ---->   1    6  !2
a1 ------------> 0  3  !2
a2 ---------------> 1  1  !2
a3 ------------------> 1  0
```

Prima soluzione

Applichiamo direttamente il metodo seguente. Avendo ottenuto i coefficienti in ordine inverso alla visualizzazione, non possiamo stamparli via via che li calcoliamo. Li collocheremo allora in AF.

```
10 REM CONVB   CONVERSIONE DECIMALE --> BASE B
20 :
30 INPUT "BASE ";B
40 :
50 INPUT "NUMERO ";N
60 :
70 VS=0:D=1          :REM VISUALIZZAZIONE RISULTATO
80 :
90 Q=INT(N/B)        :REM QUOZIENTE
100 R=N-(Q*B)        :REM RESTO
110 :
120 VS=R*D+VS
130 D=D*10
140 :
150 N=Q
160 IF Q>0 THEN 90
170 :
180 PRINT VS
190 GOTO 50


RUN
BASE ? 2
NUMERO ? 5
 101
```

Seconda soluzione

Cominciamo a calcolare i coefficienti di grado maggiore. Manualmente questo metodo sarebbe più lungo.

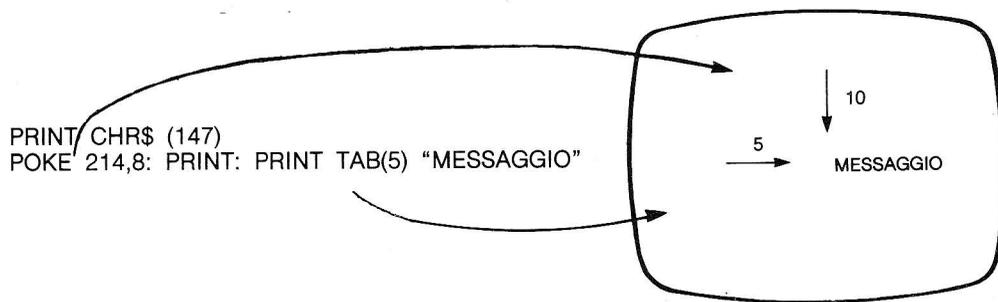
```
30 INPUT "BASE ";B
40 :
50 INPUT "NUMERO ";N
60 :
70 FOR P=6 TO 0 STEP-1
80 :X=B↑P
90 :Q=INT((N/X)+.001)
100 :N=N-X*Q
110 :PRINT Q;
120 NEXT P


RUN
BASE ? 2
NUMERO ? 5
 0  0  0  0  1  0  1
```

Indirizzamento diretto video

L'indirizzamento diretto permette di visualizzare un messaggio in un punto preciso dello schermo (senza modificarne il resto).

Il Commodore 64 non possiede istruzioni a questo scopo. Utilizzeremo dunque il puntatore di linea situato nel byte di memoria 214.
POKE 214, n. riga: PRINT ↲ posiziona il cursore sulla riga specificata + 2.
"PRINT" dopo POKE è obbligatorio.

Esempio

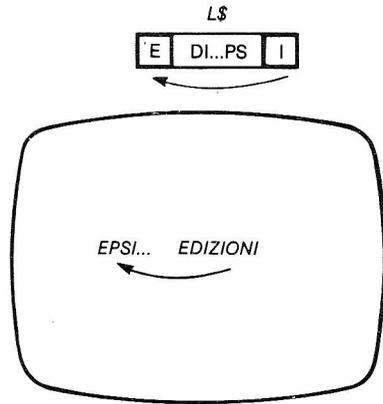
Posizionamento sulla riga 10 (8 + 2).

Scritta luminosa

Questo programma fa scorrere un messaggio sullo schermo. La velocità di scorrimento dipende dalla temporizzazione scelta.

```
10 PRINT CHR$(147)
20 L$="EDIZIONI EPSI..."
30 L=LEN(L$)
40 POKE 214,11:PRINT
50 PRINT.TAB(9)L$
60 WAIT 653,1 OR 2
70 IF PEEK(653)=2 THEN END
80 L$=RIGHT$(L$,1)+LEFT$(L$,1-1)
90 GOTO 40

READY.
```

L$

E | DI...PS | I

EPSI... EDIZIONI

L'istruzione "**WAIT** memoria, valore" provoca una pausa nel programma. Quest'ultimo riprende quando la condizione posta è soddisfatta.

- Il byte di memoria 653 contiene lo stato del tasto SHIFT/C= Control.

0 = normale
1 = SHIFT
2 = C=
4 = Control

Premendo SHIFT, la scritta si sposta; premendo C= , si arresta lo svolgimento del programma.

Immissione di un carattere da tastiera

GET carattere

Legge in permanenza un carattere da tastiera. Se non è stato battuto alcun carattere, la stringa letta (A$ nell'esempio) è vuota. Il carattere battuto da tastiera è visualizzato solo se il programma lo ha previsto (e non in modo "locale" da tastiera, come è nel caso di INPUT).

```
10 PRINT CHR$(147):REM  ABBLENCA IL VIDEO
20 GET A$:IF A$="" GOTO 20:REM  LOOP DI ATTESA
30 PRINT A$;:REM  STAMPA IL CARATTERE BATTUTO
40 GOTO 20
```

I caratteri vengono visualizzati dall'istruzione 30

Esempio: acquisizione di numeri o di lettere (secondo l'opzione scelta).

I caratteri scelti sono memorizzati in una stringa B$. Il tasto "INST/DEL" abblenca la stringa B$ e ricomincia l'immissione.
Quando l'operatore preme "RETURN" (codice 13), il programma si arresta.
Il valore numerico è collocato in A (sono accettati i decimali).
La stringa è collocata in B$ (sono accettati gli spazi).

```
100 PRINT CHR$(147)
110 INPUT "N-NUMERICO  A-ALFABETICO";S$
120 GOSUB 9060
130 PRINT B$,:IF S$="N"THEN PRINTA
140 PRINT :END
9000 REM ****************************************************
9010 :
9020 REM ***** SOTTO PROGRAMMA DI INSERIMENTO DA TASTIERA *****
```

(segue)

(*seguito*)

```
9030 :
9040 REM *****************************************************
9050 :
9060 IF S$="N"THEN MINI=48:MAXI=57:DIV=46:GOTO 9090:REM VAL.STR.NUMERICA
9070 IF S$="A"THEN MINI=65:MAXI=90:DIV=32:GOTO 9090:REM VAL.STR.ALFABETICA
9080 PRINT "ERRORE":END
9090 B$="? ":GOTO 9170:REM * AZZERAMENTO STRINGA
9100 A$="":GET A$:IF A$=""GOTO 9100:REM * ATTESA TASTIERA
9110 IF ASC(A$)=13 GOTO 9180:REM TASTO RETURN
9130 IF ASC(A$)=DIV GOTO 9160:REM SPAZIO O PUNTO
9140 IF ASC(A$)=20 GOTO 9210:REM TASTO DELETE
9140 IF ASC(A$) < MINI GOTO 9100
9150 IF ASC(A$) > MAXI GOTO 9100
9160 B$=B$+A$:REM CONCATENAZIONE DELLE STRINGHE
9170 POKE214,10:PRINT:PRINT B$:GOTO 9100:REM VISUALIZZA STRINGA
9180 B$=MID$(B$,3):IF S$="A" THEN RETURN;REM SE STRINGA ALFABETICA RITORNA
9190 A=VAL(B$)
9200 IF MID$(STR$(A),2)=B$ THEN RETURN:REM CONTROLLA IL VALORE NUMERICO
9210 A$="  ":FORI=1 TO LEN(B$):A$=A$+" ":NEXT:REM SE ERRORE ABBLENCA
9220 POKE 214,10:PRINT:PRINT A$:GOTO 9090:REM RICOMINCIA

READY.
```

PEEK (197)

Il byte di memoria 197 contiene il codice relativo alla tastiera dell'ultimo tasto battuto. Il codice rimane il medesimo qualunque sia la posizione della tastiera (SHIFT, C= , "reverse image", Minuscolo).
Il seguente esempio mostra le diverse combinazioni di codici di un tasto (ASCII, video, tastiera).
Introdotto in un programma, il comando PEEK (197) permette di definire con esattezza il tasto premuto.

Esempio

```
110 PRINT CHR$(147);
120 A=0
130 B=0:C=0:T=PEEK(197):PRINT:A=A+40
140 GET A$:IF A$="" GOTO 140
150 IF PEEK(197)<>T GOTO 130
160 IF B=3 THEN C=1
170 IF C<>0 THEN PRINT CHR$(18);A$;CHR$(146);:GOTO 190
180 PRINT A$;
190 PRINT ASC(A$),PEEK(1024+A),T
195 A=A+40:B=B+1
196 GOTO 140
```

Reverse image

Normale

A
Normal
SHIFT
A
Normal
SHIFT
A 65 1 10
193 65 10
r 176 112 10
65 129 10
193 193 10
176 240 10
Codice tastiera lettera A
32 32 60
160 96 60
160 96 60
32 160 60
160 224 60
160 224 60
Codice tastiera barra spaziatrice
B 66 2 28
194 66 28
191 127 28
66 130 28
194 194 28
191 255 28
Codice tastiera lettera B
Carattere
Codice ASCII
Codice schermo
Codice tastiera

I numeri aleatori

I numeri aleatori sono utilizzati essenzialmente per i programmi di giochi od educativi.

RND (X) per X > 0

Fornisce un numero a caso compreso tra 0 e 1 (1 escluso).

Per ottenere numeri interi fra 0 e 9, per esempio, occorre:

- Moltiplicare il numero ottenuto per 10.
- Prendere la parte intera del risultato, con la funzione INT(X).

```
10 X=RND(1)
20 Y=X*10
30 Z=INT(Y)
40 PRINT X;TAB(12);Y;TAB(24)
50 GOTO 10
READY.
RUN
 .803157845   8.03157846   8
 .562966738   5.62966738   5
```

Dà la parte intera di y (PRINT INT(2.12)→2)

RND (1) — RND (1) ✳ 10 — INT (RND (1) ✳ 10)

Per ottenere un numero fra 1 e 10, basta aggiungere 1.

```
10 FOR N=1 TO 3
20 :PRINT INT(RND(1)*10)+1
30 NEXT N
READY.
RUN
 9
 10
 2
```

RND (X) per < 0

X negativo inizializza una serie aleatoria che dipende dal valore di X. Questa serie è sempre la stessa per un valore di X.

```
10 X=RND(-2)
30 FOR I=1 TO 3
40 :PRINT RND(1);
50 NEXT I

RUN
 .865554613   .846128798   .981100118
READY.
RUN
 .865554613   .846128798   .981100118
```

Stessa sequenza

Senza 10 X = RND (—2), la serie aleatoria dopo il secondo RUN sarebbe stata diversa da quella ottenuta dopo il primo RUN.

Divisioni

Questo programma sceglie a caso 2 numeri, li visualizza, attende in risposta il loro quoziente, e verifica l'esattezza del risultato.
Il livello di difficoltà è scelto dall'inizio.

```
10 REM DIV   DIVISIONI
20 :
30 INPUT "DIVIDENDO MASSIMO (ES:1000) ";DV
40 :
50 X=INT(RND(1)*DV)+1
60 Y=INT(RND(1)*DV/10)+1
70 :
80 PRINT:PRINT "QUAL'E' IL QUOZIENTE DI ";X;"PER";Y
90 INPUT "RISPOSTA ";RP
100 IF INT(X/Y)=RP  THEN PRINT:PRINT "O.K. ":GOTO 50
110 :
120 PRINT:PRINT #IL QUOZIENTE DI ";X;"PER";Y;"E'";INT(X/Y)
130 GOTO 50
```

```
RUN
DIVIDENDO MASSIMO (ES:1000) ? 50

QUAL'E' IL QUOZIENTE DI  19 PER 4
RISPOSTA ? 4

O.K.

QUAL'E' IL QUOZIENTE DI  47 PER 4
RISPOSTA ? 11

O.K.
```

Colori

Colore di fondo e colore di contorno

• Il colore di fondo dello schermo è attribuito da: POKE 53281, colore di fondo.

• Il colore di contorno dello schermo è attribuito da: POKE 53280, colore di contorno.

Esempio

Battendo POKE 53281,1 lo sfondo dello schermo diventa bianco.

Colori differenti

0 NERO
1 BIANCO
2 ROSSO
3 TURCHESE
4 PORPORA
5 VERDE
6 BLU
7 GIALLO
8 ARANCIONE
9 MARRONE
10 ROSSO CHIARO
11 GRIGIO 1
12 GRIGIO 2
13 VERDE CHIARO
14 BLU CHIARO
15 GRIGIO 3

Il seguente programma visualizza in sequenza tutti i colori dello sfondo.

```
10 REM COLORI DI SFONDO
20 :
30 FOR C=0 TO 15
40 :POKE 53281,C
50 :FOR TP=1 TO 1000:NEXT TP
60 NEXT C
```

Colori di scrittura

Il colore della scrittura è attribuito in due modi:

- Attraverso i tasti CNTL 1, 2,... 7, 8 e C= 1, 2,... 8.

Esempio: PRINT " TURCHESE"

Provoca la visualizzazione del messaggio "TURCHESE".

- Attraverso PRINT CHR$ (codice colore)

Esempio: PRINT CHR$(159); "TURCHESE"

Tabella dei colori

Tasti	Colore	Codice
CTRL 1	NERO	CHR$ (144)
CTRL 2	BIANCO	CHR$ (5)
CTRL 3	ROSSO	CHR$ (28)
CTRL 4	TURCHESE	CHR$ (159)
CTRL 5	PORPORA	CHR$ (156)
CTRL 6	VERDE	CHR$ (30)
CTRL 7	BLU	CHR$ (31)
CTRL 8	GIALLO	CHR$ (158)
= 1	ARANCIO	CHR$ (129)
= 2	MARRONE	CHR$ (149)
= 3	ROSSO CHIARO	CHR$ (150)
= 4	GRIGIO 1	CHR$ (151)
= 5	GRIGIO 2	CHR$ (152)
= 6	VERDE CHIARO	CHR$ (153)
= 7	BLU CHIARO	CHR$ (154)
= 8	GRIGIO 3	CHR$ (155)

Nel seguente esempio, facciamo variare il colore della scrittura utilizzando prima la funzione CHR$(colore), poi i tasti CNTL e C= :

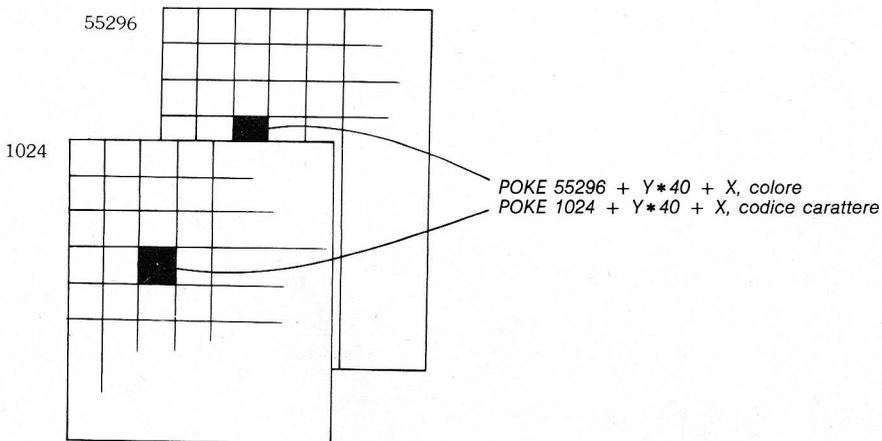
```
5 REM COLORI
6 :
10 POKE 53281,1     :REM SFONDO BIANCO
20 X$=CHR$(30 )+"VERDE"+CHR$(156)+"VIOLETTO"+CHR$(159)+"TURCHESE"
30 :
50 PRINT X$
60 :
70 Y$="▚VERDE▓VIOLETTO◣TURCHESE"
80 PRINT Y$
```

```
RUN
VERDEVIOLETTOTURCHESE
VERDEVIOLETTOTURCHESE
```

```
100 REM CL  COLORI SCRITTURA
110 :
120 POKE 53281,1          :REM SFONDO BIANCO
130 FOR N=1 TO 16
140 :READ COLORE
150 :PRINT CHR$(COLORE);"COLORE:";COLORE
160 NEXT N
180 REM---------CODICI COLORI
190 DATA 144,5,28,159,156,30,31,158
200 DATA 129,149,150,151,152,153,154,155
```

Indirizzamento grafico dello schermo (bassa risoluzione)

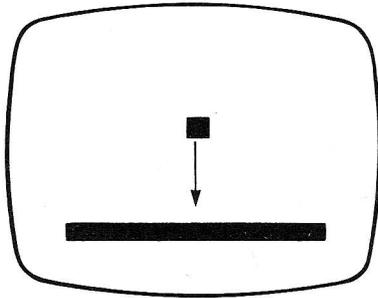
Per visualizzare a video un carattere di coordinate X,Y occorre fare:

POKE 1024+Y*40+X, codice carattere
POKE 55296+Y*40+X, colore

Esempio

```
100 PRINT CHR$(147)           :REM ABBLENCAMENTO SCHERMO
110 :
120 X=10:Y=10
130 POKE 1024+Y*40+X,102     :REM 102=CODICE DEL QUADRATO
140 POKE 55296+Y*40+X,1      :REM 1=BIANCO
```

Mattone

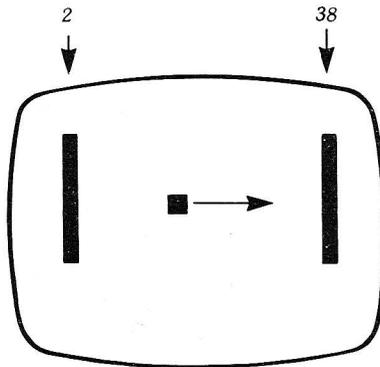
Simuliamo la caduta di un mattone partendo dall'alto dello schermo; il mattone è via via disegnato e cancellato.

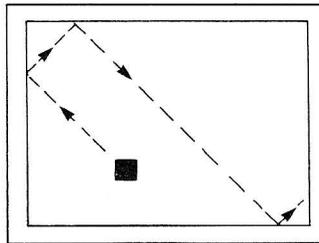
```
10 REM  MATTONE
20 :
30 PRINT CHR$(147)               :REM ABBLENCAMENTO SCHERMO
35 Y=20
40 FOR X=5 TO 20
50 :POKE 1024+X+Y*40,102         :REM 102:CODICE CARATTERE
60 :POKE 55296+X+Y*40,4          :REM CODICE COLORE
70 NEXT X
80 REM-------------------------CADUTA MATTONE
90 X=10
100 FOR Y=1 TO 19
105 :POKE 1024+X+(Y-1)*40,32     :REM ABBLENCAMENTO
110 :POKE 1024+X+Y*40,102        :REM NUOVA POSIZIONE
115 :POKE 55296+X+Y*40,1
117 :FOR TP=1 TO 20:NEXT TP      :REM TEMPORIZZAZIONE
150 NEXT Y
```

Cielo stellato multicolore

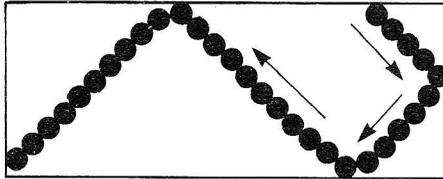
```
10 REM CIEL
100 REM  CIELO STELLATO MULTICOLORE
110 :
120 PRINT CHR$(147)
130 POKE 53281,1             :REM SFONDO BIANCO
140 FOR E=1 TO 50
150 :X=RND(1)*40
160 :Y=RND(1)*24
170 :C=RND(1)*15
180 :POKE 1024+Y*40+X,102:REM 102:CODICI DEL QUADRATO
190 :POKE 55296+Y*40+X,C     :REM COLORE
200 NEXT E
210 REM------------ COLORE DI FONDO
220 :
230 FOR TP=1 TO 3000:NEXT TP
240 C=RND(1)*15:POKE 53281,C
250 GOTO 230
```

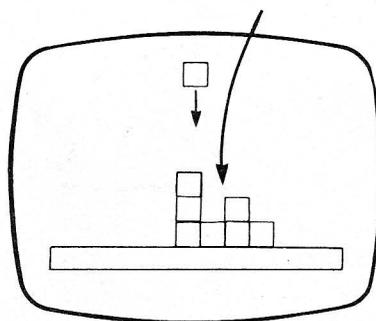
La palla che rimbalza

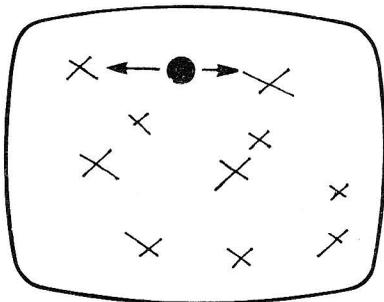
Questo programma fa rimbalzare una palla fra due muri.

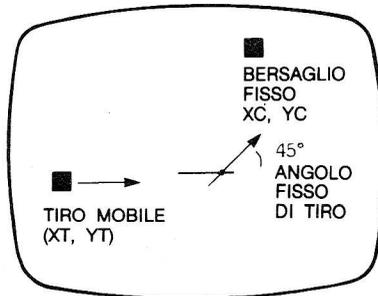
```
100 REM  PARI   PALLA RIMBALZANTE
110 :
120 PRINT CHR$(147)  :REM CANCELLAZIONE VIDEO
130 POKE 53281,1          :REM COLORE DI FONDO
140 REM---------------------CONTORNO
150 X1=2:X2=38
160 FOR Y=2 TO 10
170 :POKE 1024+Y*40+X1,102:POKE 1024+Y*40+X2,102
180 :POKE 55296+Y*40+X1,2:POKE 55296+Y*40+X2,2
190 NEXT Y
200 REM---
210 X=5:Y=5:VELO=1
220 XA=X:YA=Y                         :REM POSIZIONE PRECEDENTE
230 :
240 X=X+VELO:IF X<X1+1 OR X=>X2  THEN 310
250 :
260 POKE 1024+YA*40+XA,32             :REM CANCELLAZIONE
270 POKE 1024+Y*40+X,81:POKE 55296+Y*40+X,2
280 XA=X:YA=Y
290 GOTO 240
300 :
310 VELO=-VELO
320 GOTO 240
```

Questo fa rimbalzare la palla fra quattro muri.

```
100 REM           BILIARDO
110 :
120 PRINT CHR$(147):POKE 53281,1
130 XB=2:XH=38
140 YB=2:YH=23
150 REM------------------------------------- QUADRO
160 FOR X=XB TO XH
170 : POKE 1024+YB*40+X,102:POKE 1024+YH*40+X,102
180 :POKE 55296+YB*40+X,2:POKE 55296+YH*40+X,2
190 NEXT X
200 FOR Y=YB TO YH
210 :POKE 1024+Y*40+XB,102:POKE 1024+Y*40+XH,102
220 :POKE 55296+Y*40+XB,2:POKE 55296+Y*40+XH,2
230 NEXT  Y
240 REM---------------------
250 XP=XB+INT(RND(1)*6)+1:YP=YB+INT(RND(1)*6)+1
260 VX=1:VY=1
270 :
280 IF XP<XB+1 OR XP=>XH THEN VX=-VX:GOTO 340
290 IF YP<YB+1 OR YP=>YH THEN VY=-VY:GOTO 340
300 :
310 POKE 1024+YP*40+XP,81:POKE 55296+YP*40+XP,4
320 FOR TP=1 TO 20:NEXT TP
330 POKE 1024+YP*40+XP,32
340 XP=XP+VX:YP=YP+VY
350 GOTO 280

READY.
```

Tappezzeria

```
100 REM  TAPPEZZERIA
110 :
120 PRINT CHR$(147):POKE 53281,1
130 XB=2:XH=38:YB=2:YH=23            :REM QUADRO
140 REM----
150 XP=5:YP=6                        :REM PUNTO DI PARTENZA
160 VX=1:VY=1
170 CL=3                             :REM COLORE
180 :
190 IF XP<XB+1 OR XP=>XH THEN VX=-VX:CL=RND(1)+10+3:GOTO 230
200 IF YP<YB+1 OR YP=>YH THEN VY=-VY:GOTO 230
210 :
220 POKE 1024+YP*40+YP,81:POKE 55296+YP*40,CL
230 XP=XP+VX:YP=YP+VY
240 GOTO 190
```

Mucchio di cubi

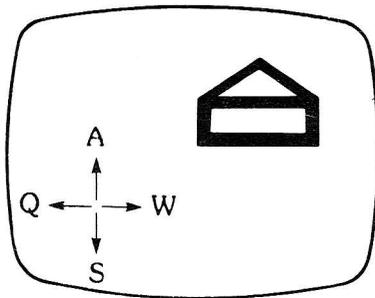
```
100 REM  MUCU   MUCCHIO DI CUBI
110 :
120 REM  I CUBI CADONO DALL'ALTO DEL    VIDEO
130 REM  RIMBALZANO E FORMANO UN MUCCHIO MULTICOLORE
140 :
150 PRINT CHR$(147):POKE 53281,1
160 PRINT CHR$(155)
170 REM--------------DISEGNO DELLA BASE
180 FOR X=1  TO 38
190 :POKE 1024+20*40+X,102:POKE 55296+20*40+X,2
200 NEXT X
210 REM------------------------- CADUTA DEI CUBI
220 X=20:Y=1:C=INT(RND(1)*8)+2
230 :
240 D=INT(RND(1)*2):IF D=0 THEN D=-1   : REM  DESTRA O SINISTRA ?
250 :
260 IF PEEK(1024+Y*40+X)<>32 THEN 300
270 POKE 1024+(Y-1)*40+X,32
280 POKE 1024+Y*40+X,81:POKE 55296+Y*40+X,C:Y=Y+1:GOTO 260
290 :
300 IF PEEK(1024+Y*40+X+D)<>32 THEN 220
310 POKE 1024+(Y-1)*40+X,32:X=X+D
320 POKE 1024+Y*40+X,81:POKE 55296+Y*40+X,C
330 FOR TP=1 TO 10:NEXT TP
340 Y=Y+1
350 IF Y>24 THEN 220
360 GOTO 260
```

Meteoriti

```
100 REM  METEORITI
110 :
120 REM  UN DISCO SI SPOSTA ATTRAVERSO DELLE METEORITI
130 REM  UTILIZZARE 'A' E 'S' PER SPOSTARSI
140 :
150 XV=15:YY=2:AX=XV:AY=YV
160 PRINT CHR$(147):POKE 53281,1    :REM SFONDO BIANCO
170 PRINT CHR$(154)                 :REM SCRITTURA BLU
180 REM----
190 PRINT:PRINT:PRINT:PRINT
200 FOR L=1 TO 10000
210 :X=RND(1)*18+1:PRINT SPC(X);"X";TAB(18);
215 :X=RND(1)*18+1:PRINT SPC(X);"X"
220 :IF PEEK(1024+YV*40+XV)<>24 THEN 270   :REM TEST COLLISIONE
230 :POKE 214,21:PRINT:PRINT "SCORE:";L;
240 :FOR TP=1 TO 3000:NEXT TP
250 :GOTO 150
260 :
270 :POKE 1024+(AY-1)*40+AX,32      :REM CANCELLAZIONE
280 :POKE 1024+YV*40+XV,81:POKE 55296+YV*40+XV,2
290 :AX=XV:AY=YV
300 :
310 :C=PEEK(197)                    :REM TEST TASTIERA
320 :IF C=64 THEN 350
330 :IF C=10 THEN IF XV>2 THEN XV=XV-1
340 :IF C=13 THEN IF XV<36 THEN XV=XV+1
350 NEXT L
```

Tiro al piattello

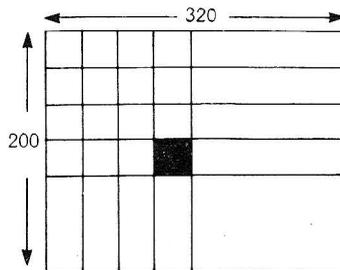
Ecco un programma di tiro. Il bersaglio è fisso ed il tiro è mobile (CHAR per esempio).
L'angolo di tiro è di 45 gradi; potrebbe essere cambiato modificando la velocità VE.

```
100 REM  TIPI   TIRO AL PIATTELLO
110 :
120 INPUT "LIVELLO (1,2,3,..) ";HV
130 :
140 PRINT CHR$(147):POKE 53281,3
150 DP=0
160 XC=INT(RND(1)*5)+30:YC=5
170 POKE 1024+YC*40+XC,102:POKE 55296+YC*40+XC,2
180 VT=-1                         :REM VELOCITA' DI TIRO
190 XT=1:YT=10+NV                 :REM COORDINATE DEL TIRO
200 :
210 POKE 1024+AY*40+AX,32         :REM CANCELLA LA POSIZIONE PRECEDENTE
220 POKE 1024+YT*40+XT,102:POKE 55296+YT*40+XT,2
230 AX=XT:AY=YT
240 IF YT<YC THEN GOTO 350
250 IF XT>37 THEN GOTO 350
260 IF XT=XC AND YC=YT THEN POKE 214,10:PRINT:PRINT "COLPITO":GOTO 370
270 :
280 GET X$:IF X$<>"" THEN DP=1      :REM PARTENZA PER IL TIRO
290 IF DP=0 THEN 330
300 :
310 YT=YT+VT
320 :
330 XT=XT+1:GOTO 210
340 :
350 POKE 214,0:PRINT:PRINT "MANCATO";
360 :
370 FOR TP=1 TO 3000:NEXT TP:GOTO 140
```

Teleschermo

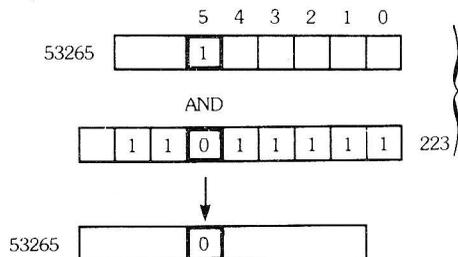
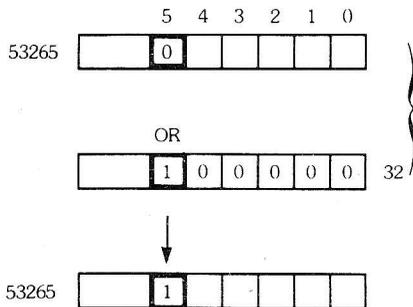
Principio: battendo i tasti Q,W,A,S spostate un "punto" che lascia una "traccia" al suo passaggio, il che vi permette di disegnare "manualmente".
Se premete "L", lo spostamento si effettua senza lasciare traccia, permettendo così di disegnare una figura in più parti. "E" permette di cancellare dei punti illuminati.

Migliorie possibili: registrazione del disegno in una tabella per duplicazione in scala.

```
100 REM TELESCHERMO
110 :
120 REM      3 MODI: DISEGNO/CANCELLAZIONE/NEUTRO
130 :
140 PRINT CHR$(147)                    :REM CANCELLAZIONE SCHERMO
150 POKE 214,21:PRINT:PRINT "T:TOGLIERE/ M:METTERE/ C:CANCELLARE";
160 POKE 214,22:PRINT:PRINT "QWAS PER SPOSTARSI";
170 X=10:Y=10
180 :
190 REM-------- LAMPEGGIAMENTO CURSORE
200 GET C$:IF C$<>"" THEN 250
210 POKE 1024+Y*40+X,102:POKE 55296+Y*40+X,1
220 POKE 1024+Y*40+X,32
230 GOTO 180
240 :
250 IF L=1 THEN IF T=1 THEN POKE 1024+Y*40+X,102
260 IF L=0 THEN POKE 1024+Y*40+X,102
270 :
280 IF C$="Q" THEN IF X>1 THEN X=X-1    :REM SINISTRA
290 IF C$="W" THEN IF X<39 THEN X=X+1   :REM DESTRA
300 IF C$="A" THEN IF Y<22 THEN Y=Y+1   :REM BASSO
310 IF C$="S" THEN IF Y>2 THEN Y=Y-1    :REM ALTO
320 :
330 IF C$="T" THEN L=1
340 IF C$="M" THEN L=0
350 IF C$="C" THEN L=2
360 :
370 T=0: IF PEEK(1024+Y*40+X)=102 THEN T=1
380 GOTO 200
```

Grafica ad alta risoluzione

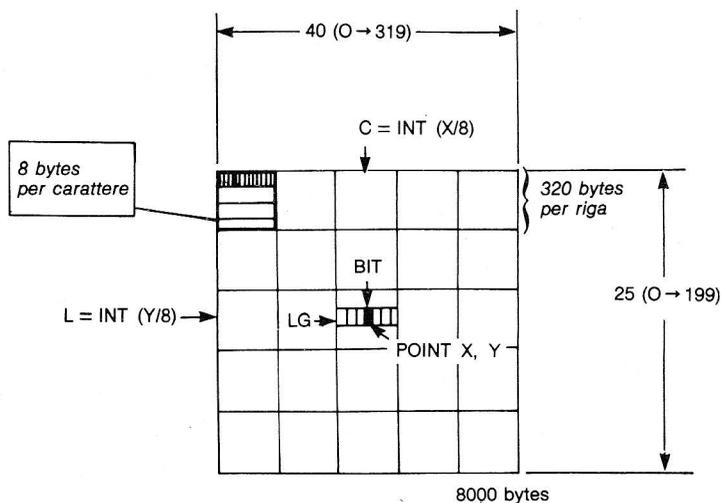
Lo schermo è diviso in 320 * 200 = 64.000 punti.
In versione base (senza "TOOL"), la programmazione della grafica ad alta risoluzione è fastidiosa.

• **Occorre posizionare il commutatore grafico a 1** (bit 5 del byte 53265):

Si fa POKE 53265, PEEK (53265) OR 32
La commutazione in senso contrario si compie con:

POKE 53265, PEEK (53265) AND 223

• **La bit-map (mappa grafica bit) deve essere rimessa a zero (8000 bytes).**

Questa operazione è molto lunga da eseguire in BASIC.
Il seguente programma (disegno di rette) utilizza una routine in linguaggio macchina.

• **Posizioniamo a 1 il bit del punto da accendere.**

La bit-map grafica è così organizzata:

Il calcolo del n. di bytes si compie utilizzando variabili intermedie: L, C e LG.
La posizione del bit all'interno del byte si ottiene con: **BIT = X AND 7.**
Ma essendo i bits sistemati da sinistra a destra, occorre fare: **BIT = 7 — (X AND 7)**

C = INT(X/8)
L = INT(Y/8)
LG = Y AND 7
OCT = AGRA + L∗320 + 8∗C + LG
BIT = 7 — (X AND 7)

Modificando un bit, non bisogna alterare gli altri del byte. Si effettua un "OPPURE" fra il vecchio contenuto e il bit da modificare:
POKE OCT, PEEK (OCT) OR (2↑P).
In realtà, per economizzare il tempo di calcolo di 2↑P, abbiamo calcolato le potenze di 2 in una tabella P( ).

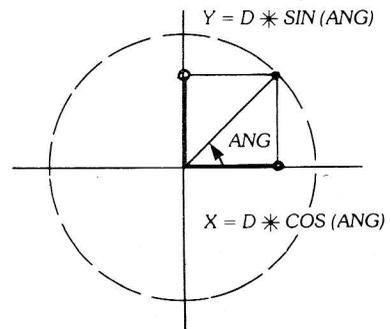
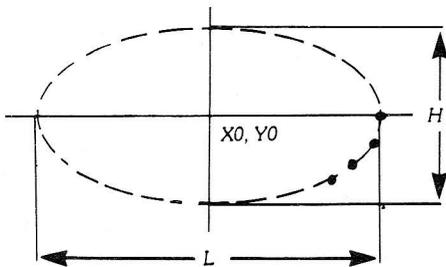
Nota: in fase di messa a punto, fare GOTO 370 per fare apparire il messaggio di errore (che non viene visualizzato in modo grafico).

Disegno di cerchi o ellissi

Un cerchio può essere tracciato punto per punto oppure per segmenti di rette. Naturalmente, viene tracciato molto più lentamente da un'istruzione BASIC. Calcoliamo le coordinate X e Y di un cerchio di raggio R con:

$$X = R * COS\ (ANG)$$
$$Y = R * SIN\ (ANG)$$

Facendo variare ANG da 0 a 6.28 radianti.
Per una ellisse, X ed Y sono dati da:

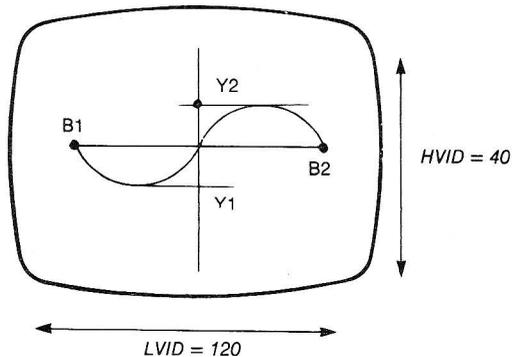
$X = (L/2) * COS\ (ANG)$
$Y = (H/2) * SIN\ (ANG)$

```
100 REM DCER    DISEGNO DI UN'ELLISSE
110 :
120 PRINT CHR$(147)                     :REM CANCELLAZIONE VIDEO
130 AGRA=8192:POKE 53272,PEEK(53272) OR 8
140 POKE 53265,PEEK(53265) OR 32    :REM COMMUTAZIONE GRAFICO
150 FOR I=AGRA TO AGRA+7999:POKE I,0:NEXT I  :REM CORRENTE MAPPA-BIT GRAFICA
160 FOR I=1024 TO 1024+999:POKE I,3:NEXT I  :REM COLORE VIDEO
170 :
180 FOR I=0 TO 7:P(I)=2↑I:NEXT I    :REM POTENZA 2
190 REM-----------------------DISEGNO DI UN'ELLISSE (O DI UN CERCHIO)
200 X0=100:Y0=100           :REM CENTRO
210 L1=80:H1=50             :REM LUNGHEZZA/ALTEZZA
220 FOR ANG=0 TO 6.29 STEP 6.28/40
230 :X=X0+(L1/2)*COS(ANG)
240 :Y=Y0+(H1/2)*SIN(ANG)
250 :GOSUB 300
260 NEXT ANG
270 GOTO 360
280 REM-----------------------ROUTINE DI DISEGNO DI UN PUNTO
290 REM INGRESSI:X,Y
300 C=INT(X/8):L=INT(Y/8):LG=Y AND 7
310 OCT=AGRA+L*320+8*C+LG
320 BIT=7-(X AND 7)
330 POKE OCT,PEEK(OCT) OR P(BIT)
340 RETURN
350 REM--------------- RIPRISTINO
360 GET X$:IF X$="" THEN 360
370 POKE 53272,PEEK(53272) AND 247
380 POKE 53265,PEEK(53265) AND 223
390 END
```

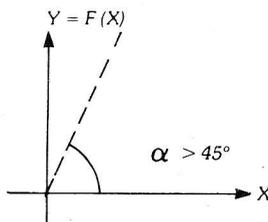
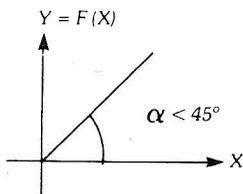
Disegno di curve sullo SCHERMO

Ecco un programma che disegna la curva con equazione Y = f(X).
Cerchiamo il massimo ed il minimo di Y: Y1 e Y2 e calcoliamo le scale per X e Y, in modo da sfruttare al meglio la dimensione dello schermo.

```
100 REM DCURV               DISEGNO DI CURVE AUTOMATICHE SUL VIDEO
110 :
120 LVID=140:HVID=100           :REM      DIMENSIONI SCHERMO (LUNGHEZZA
130 INPUT "LIMITE 1 ";B1                                       /ALTEZZA)
140 INPUT "LIMITE 2 ";B2
150 INPUT "INCREMENTO ";INC
160 :
170 PRINT CHR$(147)
180 AGRA=8192:POKE 53272,PEEK(53272) OR 8
190 POKE 53265,PEEK(53265) OR 32
200 FOR I=AGRA TO AGRA+7999:POKE I,0:NEXT I
210 FOR I=1024 TO 1024+999:POKE I,3:NEXT I
220 FOR I=1024 TO 1024+999:POKE I,3:NEXT I
230 FOR I=0 TO 7:P(I)=2↑I:NEXT I
240 REM-------------------------RICERCA MIN/MAX
250 X=B1:GOSUB 570:Y1=Y:Y2=Y
260 FOR X=B1 TO B2 STEP INC
270 :GOSUB 570
280 :IF Y<Y1 THEN Y1=Y
290 :IF Y>Y2 THEN Y2=Y
300 NEXT X
310 PX=LVID/(B2-B1)            :REM SCALA X
320 PY=HVID/(Y2-Y1)            :REM SCALA Y
330 REM-----------------------------ASSE Y
340 IF B2<0 OR B1>0 THEN 380
350 SX=ABS(B1)*PX
360 FOR SY=0 TO HVID:GOSUB 500:NEXT SY
370 REM------------------------ ASSE X
380 IF Y2<0 OR Y1>0 THEN 420
390 SY=HVID-ABS(Y1)*PY
400 FOR SX=0 TO LVID:GOSUB 500:NEXT SX
410 REM--------------------- CURVA
420 FOR X=B1 TO B2 STEP INC
430 :GOSUB 570
440 :SX=(X-B1)*PX
450 :SY=HVID-(Y-Y1)*PY
460 :GOSUB 500
470 NEXT X
480 GOTO 610
490 REM----------- INSERIMENTO:SX,SY
500 C=INT(SX/8):L=INT(SY/8):LG=SY AND 7
510 OCT=AGRA+L*320+8*C+LG
520 BIT=7-(SX AND 7)
530 POKE OCT,PEEK(OCT) OR P(BIT)
540 RETURN
570 REM------------------- CURVA DA      COMPLETARE
580 Y=X*X-4
590 RETURN
600 REM---------------------RIPRISTINO
610 GET C$:IF C$="" THEN 610
620 POKE 53272,PEEK(53272) AND 247
630 POKE 53265,PEEK(53265) AND 223
```

HVID = 40

LVID = 120

Disegno di una retta

Per tracciare una retta, potremmo calcolare Y = F(X) facendo variare X.

Ma per una retta di inclinazione superiore a 1, il tracciato sarebbe discontinuo. Bisognerebbe allora calcolare X = F(Y) facendo variare Y.
Ci proponiamo di calcolare X = F(D) e Y = F(D) facendo variare D. Non avremo così discontinuità.

Disegno di rette

```
100 REM DISL   DISEGNO DI LINEE
110 :
120 REM   CON CORRENTE BIT-MAP GRAFICA
130 :
140 REM DISEGNO DI RETTA CON QUALSIASI  INCLINAZIONE
150 :
160 PRINT CHR$(147)
170 FOR I=832 TO 855:READ A:POKE I,A:NEXT I:SYS832
180 :
190 DATA 169,32,133,98,160,0,132,97,169
200 DATA 0,145,97,200,208,251,230,98,165,98,201,64,208,241,96
210 :
220 AGRA=8192:POKE 53272,PEEK(53272) OR 8
230 :
240 POKE 53265,PEEK(53265) OR 32        :REM COMMUTAZIONE GRAFICA
250 :
260 FOR I=1024 TO 1024+999:POKE I,3:NEXT I
270 :
280 FOR I=0 TO 7:P(I)=2↑I:NEXT I
290 REM---------------------------------------- STUDIO (FUNZIONE)
300 X1=10:X2=100:Y1=10:Y2=100:GOSUB 360
310 X1=X2:Y1=Y2:X2=X1+40:Y2=Y1:GOSUB 360
320 GOTO 480
330 REM------------------ DISEGNO DI UNA RETTA
340 REM INGRESSI: X1,X2  Y1,Y2
350 :
360 DX=X2-X1:DY=Y2-Y1:D=SQR(DX*DX+DY*DY)
370 CX=DX/D:CY=DY/D    :REM INCLINAZIONI
380 :
390 FOR DD=0 TO D
400 X=X1+DD*CX:Y=Y1+DD*CY
410 C=INT(X/8):L=INT(Y/8):LG=Y AND 7
420 OCT=AGRA+L*320+8*C+LG
430 BIT=7-(X AND 7)
440 POKE OCT,PEEK(OCT) OR P(BIT)
450 NEXT DD
460 RETURN
470 REM------------------ RIPRISTINO
480 GET X$:IF X$="" THEN 480
490 POKE 53272,PEEK(53272) AND 247
500 POKE 53265,PEEK(53265) AND 223
510 REM--------
520 REM  PER METTERE SU OPEN, IN CASO DI ERRORE, FARE GOTO 490
530 REM   PER VISUALIZZARE IL MESSAGGIO DI ERRORE
```

```
10 REM GEO INTERROGAZIONE GEOGRAFIA
20 :
30 :
35 DIM XV(20),YV(20),C$(20)
40 PRINT CHR$(147)
50 BASE=4096*2:POKE 53272,PEEK(53272) OR 8
60 :
70 POKE 53265,PEEK(53265) OR 32      :REM COMMUTAZIONE GRAFICA
80 :
90 FOR I=BASE TO BASE+7999:POKE I,0:NEXT I
100 :
110 FOR I=1024 TO 1024+999:POKE I,3:NEXT I
120 :
130 FOR I=0 TO 7:P(I)=2^I:NEXT I
140 REM-------------------------------
150 :
200 READ X1,Y1
210 :
220 READ X2,Y2
225 IF X2=999 THEN 280
230 GOSUB 800
240 X1=X2:Y1=Y2
250 GOTO 220
260 :
270 REM-----------LETTURA CITTA' IN C$()
280 V=0
285 :
290 READ X,Y,C$:IF X=999 THEN NV=V:GOTO 1150
300 V=V+1
310 XV(V)=X:YV(V)=Y:C$(V)=C$
320 GOTO 290
790 REM------------DISEGNO DI UNA RETTA
792 REM IMMISSIONI: X1,X2  Y1,Y2
795 :
800 DX=X2-X1:DY=Y2-Y1:D=SQR(DX*DX+DY*DY)
810 CX=DX/D:CY=DY/D
820 FOR DD=1 TO D
830 X=X1+DD*CX:Y=Y1+DD*CY
1000 C=INT(X/8):CL=INT(Y/8):LG=Y AND 7
1010 OCT=BASE+CL*320+8*C+LG
1020 BIT=7-(X AND 7)
1030 POKE OCT,PEEK(OCT) OR P(BIT)
1050 NEXT DD
1060 RETURN
1100 REM=======================
1150 V=INT(RND(1)*NV)+1
1155 X=XV(V):Y=YV(V):
1159 :
1160 C=INT(X/8):CL=INT(Y/8):LG=Y AND 7
1165 OCT=BASE+CL*320+8*C+LG
1170 BIT=7-(X AND 7)
1175 POKE OCT,PEEK(OCT) OR P(BIT)
1180 :
1185 FOR TP=1 TO 2000:NEXT TP
1190 GOSUB 1250
1194 :
1195 POKE 214,21:PRINT
1196 PRINT "                         "
1197 PRINT "                         "
```

(segue)

(*seguito*)

```
1198 POKE 214,21:PRINT
1200 INPUT "QUALE CITTA' ";C$
1201 FOR I=1 TO LEN(C$)
1202 :IF ASC(MID$(C$,I,1))=32 THEN 1204
1203 NEXT I
1204 C$=LEFT$(C$,I-1)
1205 IF C$="FINE" THEN GOSUB 1260:END
1210 IF C$=C$(V) THEN PRINT "O.K.."
1220 IF C$<>C$(V) THEN PRINT "NO, E' ";C$(V)
1230 FOR TP=1 TO 2000:NEXT TP
1235 GOSUB 1300
1236 POKE OCT,0
1240 GOTO 1150
1250 REM---------- COMMUTAZIONE TESTO
1260 POKE 53272,PEEK(53272) AND 247
1270 POKE 53265,PEEK(53265) AND 223
1280 RETURN
1290 REM---------- COMMUTAZIONE GRAFICA
1300 POKE 53272,PEEK(53272) OR 8
1310 POKE 53265,PEEK(53265) OR 32
1320 RETURN
1490 REM---------- PIANTA DI FRANCIA
1500 DATA 155,2
1510 DATA 223,34,   224,38
1520 DATA 217,49,   211,77,   205,85
1530 DATA 204,90,   205,93,   211,88
1540 DATA 215,93,   215,113,  216,119
1550 DATA 224,130,  212,140,  203,145
1560 DATA 196,139,  184,136,  172,141
1570 DATA 168,148,  166,156,  104,141
1580 DATA 112,117,  115,94,   109,86
1590 DATA 100,74,   98,66,    88,59
1600 DATA 77,51,    72,50,    72,45
1610 DATA 70,39,    75,37,    82,38
1620 DATA 89,35,    98,38,    105,40
1630 DATA 101,33,   99,27,    99,22
1640 DATA 99,18,    106,21,   112,21
1650 DATA 117,27,   122,29,   129,22
1660 DATA 140,18,155,3
1700 DATA 999,999
1710 :
1800 DATA 152,45,PARIS
1810 DATA 162,15,LILLE
1820 DATA 172,36,REIMS
1830 DATA 183,70,DIJON
1840 DATA 203,71,BELFORT
1850 DATA 179,103,LYON
1860 DATA 185,126,AVIGNON
1870 DATA 196,135,MARSEILLE
2000 DATA 999,999,ZZZ
```

Teleschermo

Funzioni dei tasti

H	accende un punto a destra
SHIFT/H	accende un punto a sinistra
C= /H	spegne un punto a destra
V	accende un punto discendendo
SHIFT/V	accende un punto salendo
C= /V	spegne un punto discendendo
RETURN	ritorna all'inizio della riga seguente
CLR HOME	ritorna in alto a sinistra
INST DEL	cancella lo schermo grafico
F	ritorna in modo normale

La messa a zero della carta grafica si effettua alla riga 160. Essa inizializza e lancia un sottoprogramma in linguaggio macchina.
Per aggiungere altre funzioni a questo programma (disegno di rette, cerchi, ecc.):

- scrivere in DATA i valori ASCII dei tasti utilizzati;
- aumentare la variabile "direttiva" alla riga 130;
- scrivere le istruzioni corrispondenti fra le righe 690 e 2000;
- aggiungere le direzione (ON A GOTO n. riga 1, n. riga 2, ...) alla riga 270.

```
100 REM ************************************
110 REM ***  TELESCHERMO GRAFICO   ************
120 REM ************************************
130 DIRECTIVE=10
140 DIMTB%(DIRECTIVE)
150 REM * AZZERAMENTO CARTA GRAFICA *
160 FOR I=832 TO 855:READA:POKE I,A:NEXT:SYS832
170 FOR I=1 TO DIRECTIVE:READ TB%(I):NEXT
180 SCHERMO=8192:REM * INDIRIZZO GRAFICO *
190 GOSUB 2260:REM * COMMUTAZIONE GRAFICA *
200 FOR I=0 TO 7:POIDS(I)=2^I:NEXT I:REM * POIDS BIT *
210 X=0:Y=0:REM * INIZIALIZZAZIONE PARTENZA *
220 GET A$:IF A$="" GOTO 220:REM * ATTENDE ISTRUZIONI
230 REM * CONTROLLO ISTRUZIONI
240 A=ASC(A$):FOR I=1 TO DIRECTIVE:IF TB%(I)<>A THEN NEXT:GOTO 220
250 A=I:I=DIRECTIVE:NEXT:REM * TERMINA IL LOOP
260 REM * GOTO ISTRUZIONI SEGUENTI
270 ON A GOTO 510,530,550,570,590,610,630,650,670,690
```

(*segue*)

(seguito)

```
500 REM *** SCRIVE UN PUNTO SULLA DESTRA ***
510 GOSUB 2010:GOSUB 2190:GOTO 220
520 REM *** SCRIVE UN PUNTO SULLO SINISTRA ***
530 GOSUB 2150:GOSUB 2190:GOTO 220
540 REM *** CANCELLA UN PUNTO A DESTRA ***
550 GOSUB 2150 :GOSUB 2010:GOTO 220
560 REM *** SCRIVE UN PUNTO DISCENDENDO ***
570 GOSUB 2030:GOSUB 2190:GOTO 220
580 REM *** SCRIVE UN PUNTO SALENDO ***
590 GOSUB 2070:GOSUB 2190:GOTO 220
600 REM *** CANCELLA UN PUNTO DISCENDENDO ***
610 GOSUB 2150:GOSUB 2030:GOTO 220
620 REM *** RITORNA ALLA RIGA ***
630 X=0:GOSUB 2030:GOTO 220
640 REM *** RITORNA IN ALTO A SINISTRA ***
650 GOTO 210
660 REM *** DELETE ***
670 RUN
680 REM *** FINE ***
690 GOSUB 2230:END
2000 REM * CONTROLLO PUNTATORI X,N
2010 X=X+1:IF X<319 THEN RETURN
2020 X=0
2030 Y=Y+1:IF Y>199 THEN Y=199
2040 RETURN
2050 X=X-1:IF X=0 OR X>0 THEN RETURN
2060 X=319
2070 Y=Y-1:IF Y=0 OR Y>0 THEN RETURN
2080 Y=0:RETURN
2090 REM * CALCOLA IL BYTE *
2100 X1=INT(X/8):Y1=INT(Y/8)
2110 OCTET=SCHERMO+(X1*8)+(Y1*320)+(Y AND 7)
2120 BIT=X AND 7:REM * POSIZIONE DEL BIT NEL BYTE *
2130 RETURN
2140 REM * SPEGNE UN PUNTO *
2150 GOSUB 2100
2160 POKE OCTET,PEEK(OCTET) AND (255-POIDS(7-(BIT AND 7)))
2170 RETURN
2180 REM * ACCENDE UN PUNTO *
2190 GOSUB 2100
2200 POKE OCTET,PEEK(OCTET) OR POIDS(7-(BIT AND 7))
2210 RETURN
2220 REM * RITORNA NORMALE *
2230 POKE 53272,PEEK(53272)AND 247
2240 POKE 53265,PEEK(53265)AND 223
2250 POKE 650,0:RETURN
2260 PRINT CHR$(147):REM * CANCELLA LO SCHERMO *
2270 POKE 53272,PEEK(53272) OR 8:REM * PASSA IN MODO GRAFICO *
2280 POKE 53265,PEEK(53265) OR 32
2290 POKE 650,128
2300 REM * CODICE COLORE *
2310 REM POKE 53280,   :REM COLORE DEL QUADRO
2320 REM POKE 53281,   :REM COLORE DELLO SFONDO 0
2330 REM POKE 53282,   :REM COLORE DELLO SFONDO 1
2340 REM POKE 53283,   :REM COLORE DELLO SFONDO 2
2350 REM POKE 53284,   :REM COLORE DELLO SFONDO 3
2360 RETURN
2500 REM ***********************************
2510 REM ** CODICI 'LINGUAGGIO MACCHINA'  **
2520 REM ***********************************
```

(segue)

(seguito)

```
2530 DATA 169,32,133,98,160,0,132,97,169
2540 DATA 0,145,97,200,208,251,230,98,165,98,201,64,208,241,96
2550 REM **********************************
2560 REM ** CODICI 'ASCII DEI TASTI'     ***
2570 REM **********************************
2580 DATA 72,200,180,86,214,190,13,19,20,70
```

Mini-interprete di loops

Per chiarire come funzioni un "Interprete Basic", abbiamo scritto in DATA uno pseudo-programma.

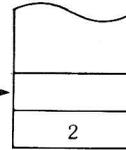
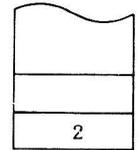
```
10 REM INTER   MINI-INTERPRETE DI LOOP
20 :
30 REM------------- PSEUDO PROGRAMMA INTERPRETE
40 DATA RIPETI 2
50 DATA  PRINT "GRANDE LOOP"
60 DATA  RIPETI 3
70 DATA   PRINT "  LOOP INTERNO"
80 DATA  ANCORA
90 DATA ANCORA
100 DATA *
110 REM--------------------------------
120 FOR I=1 TO 100
130 :READ X$:IF X$="*" THEN NC=I-1:GOTO 170
140 :CD$(I)=X$
150 NEXT I
160 :
170 PL=1    :REM OPERATORE RIGA IN CORSO
180 :
190 LG$=CD$(PL)
200 C$=LEFT$(LG$,4)
210 IF C$="RIPE" THEN 330
220 :
230 IF C$="ANCO" THEN GOTO 370
240 :
250 PRINT PL;LG$
270 PL=PL+1
280 IF PL>NC THEN STOP
290 GOTO 190
300 REM------------------------------------ RIPETE
310 REM   PR--> OPERATORE COLONNE RP() ]CONTENENTE N.DI LINEE DOPO RIPETI
320 REM                        NB() CONTENENTE N.DI LOOP
330 FOR P=1 TO LEN(LG$)
332 :IF MID$(LG$,P,1)=" " THEN 340     :REM SPAZIO
334 NEXT P
336 P=0
340 PR=PR+1:RP(PR)=PL+1:NB(PR)=VAL(RIGHT$(LG$,LEN(LG$)-P))
350 PL=PL+1:GOTO 190
360 REM------------------------------------ ANCORA
370 NB(PR)=NB(PR)-1:IF NB(PR)>0 THEN PL=RP(PR):GOTO 190
380 PR=PR-1:PL=PL+1:GOTO 190
```

Tabella CD$ ( )

1	RIPETERE 2
2	PRINT "GRANDE LOOP"
3	RIPETERE 3
4	PRINT "LOOP INTERNO"
5	ANCORA
6	ANCORA

PL → (riga 4)

PR →

COLONNA RP( )	COLONNA NB( )
2	2

COLONNA RP( ) — *N° riga*

COLONNA NB( ) — *numero dei loop*

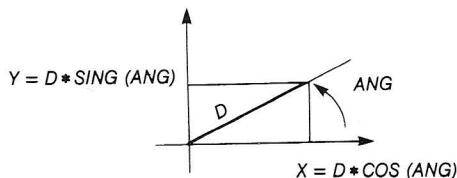
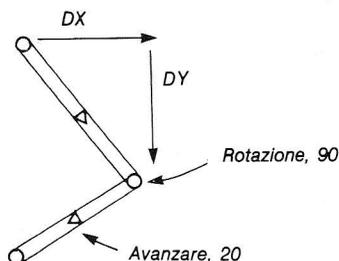
```
2 PRINT "GRANDE LOOP"
4 PRINT "  LOOP INTERNO"
4 PRINT "  LOOP INTERNO"
4 PRINT "  LOOP INTERNO"
2 PRINT "GRANDE LOOP"
4 PRINT "  LOOP INTERNO"
4 PRINT "  LOOP INTERNO"
4 PRINT "  LOOP INTERNO"
7
```

Disegno di figure

La maggior parte dei linguaggi propongono delle istruzioni grafiche dove vengono specificate le coordinate X, Y delle rette da tracciare. Il linguaggio LOGO propone delle istruzioni grafiche alquanto originali. Ne abbiamo simulate due in BASIC:

AVANZAMENTO distanza: Disegno di un segmento di lunghezza uguale a "distanza".

ROTAZIONE angolo: Cambio di direzione della linea aggiungendo "angolo".

```
100 REM DLOG     DISEGNO DI FIGURE (LOGO)
110 :
120 PRINT CHR$(147)
130 AGRA=8192:POKE 53272,PEEK(53272) OR 8
140 POKE 53265,PEEK(53265) OR 32   :REM COMMUTAZIONE GRAFICA
150 :
160 FOR I=AGRA TO AGRA+7999:POKE I,0:NEXT I
170 FOR I=1024 TO 1024+999:POKE I,3:NEXT I
180 FOR I=0 TO 7:P(I)=2↑I:NEXT I
190 REM-----------------------------------------
200 X0=100:Y0=100:CX=1:CY=0
210 READ CM$
220 IF CM$="FINE" THEN 450
230 IF CM$="ROTAZIONE" THEN READ R:ANG=ANG+R:GOSUB 270
240 IF CM$="AVANZARE" THEN READ DIST:GOSUB 350
250 GOTO 210
260 REM------------ ANGOLO
270 IF ANG=>360 THEN ANG=ANG-360
280 AR=ANG/360*2*3.14159
290 CX=COS(AR):CY=SIN(AR)
300 IF CX>-.01 AND CX<.01 THEN CX=0
310 IF CY>-.01 AND CY<.01 THEN CY=0
320 RETURN
330 REM------------------- DISEGNO DI UNA RETTA
340 :
350 FOR DD=0 TO DIST
360 X=X0+DD*CX:Y=Y0+DD*CY
370 C=INT(X/8):CL=INT(Y/8):LG=Y AND 7
380 OCT=AGRA+CL*320+8*C+LG
390 BIT=7-(X AND 7)
400 POKE OCT,PEEK(OCT) OR P(BIT)
410 NEXT DD
420 X0=X0+DIST*CX:Y0=Y0+DIST*CY
430 RETURN
440 REM----------------- RIPRISTINO
450 GET X$:IF X$="" THEN 450
460 POKE 53272,PEEK(53272) AND 247
470 POKE 53265,PEEK(53265) AND 223
480 END
490 REM--------- DISEGNO DI UN QUADRATO
500 DATA AVANZARE,30
510 DATA ROTAZIONE,90
520 DATA AVANZARE,30
530 DATA ROTAZIONE,90
540 DATA AVANZARE,30
550 DATA ROTAZIONE,90
560 DATA AVANZARE,30
570 DATA FINE
```

Mini-interprete Logo

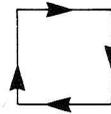
Interpretiamo in BASIC alcune istruzioni grafiche del linguaggio LOGO.

Istruzioni di base

DISTANZA 10	La distanza corrente diviene 10.
AVANZAMENTO	Avanza della distanza corrente.
AVANZAMENTO 10	Avanza di una distanza 10.
ROTAZIONE 90	La direzione è modificata di 90°.
DISTANZA + 3	La distanza è aumentata di 3.
LEVA	Interrompe il disegno.
CALA	Riprende il disegno.

Loops

Le istruzioni tra RIPETI X e ANCORA sono eseguite X volte.

```
RIPETI 4
 ROTAZIONE 90
 AVANZAMENTO
ANCORA
```

Questa sequenza disegna un quadrato.

Funzioni

Alcune "funzioni" si dichiarano in questo modo:

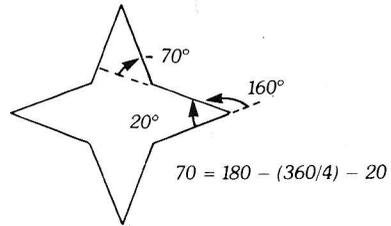
```
DEFI TRIANGOLO
RIPETI 3
 AVANZAMENTO
 ROTAZIONE 120
ANCORA
FINE
```

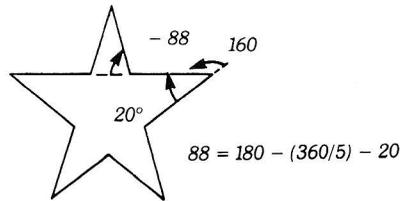
Per disegnare un triangolo, basta poi scrivere:

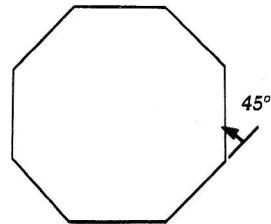
```
TRIANGOLO
```

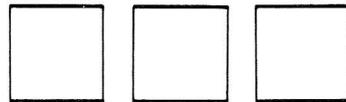
Al momento della lettura delle istruzioni, la "funzione" richiamata è rimpiazzata dalla sequenza delle istruzioni che vi sono scritte.

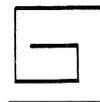
Disegno di una stella a 4 punte

```
RIPETI 4
 AVANZAMENTO 10
 ROTAZIONE 160
 AVANZAMENTO 10
 ROTAZIONE -70
ANCORA
```

Stella a 5 punte

```
RIPETI 5
 AVANZAMENTO 10
 ROTAZIONE 160
 AVANZAMENTO 10
 ROTAZIONE -88
ANCORA
```

Ottagono

```
RIPETI 8
 AVANZAMENTO 10
 ROTAZIONE 45
ANCORA
```

3 quadrati

```
RIPETI 3
 DISTANZA 5
 CALA
 QUADRATO
 LEVA
 AVANZAMENTO 15
ANCORA
```

Spirale

```
DISTANZA 5
RIPETI 6
 AVANZAMENTO
 ROTAZIONE 90
 DIS+2
ANCORA
```

Simulatore Logo

```
10 REM LOGO      SIMULATORE LOGO
15 :
20 :
30 DIM CD$(100):Y0=80:X0=60:CX=1:CY=0:DIST=20
40 :
50 :
180 REM--------------- DEFINIZIONE QUADRATO
190 DATA DEFI QUADRATO
200 DATA ABBASSARE
210 DATA RIPETERE 4
220 DATA  AVANZARE
230 DATA  ROTAZIONE 90
240 DATA ANCORA
250 DATA FINE
260 REM------------------- DEFINIZIONE TRIANGOLO
270 DATA DEFI TRIANGOLO
280 DATA RIPETERE 3
290 DATA  ROTAZIONE 120
300 DATA  AVANZARE
310 DATA ANCORA
320 DATA FINE
330 REM----------------------
340 DATA DISTANZA 30
350 DATA QUADRATO
355 :
360 DATA ALZARE
370 DATA AVANZARE 60
375 DATA ABBASSARE
380 DATA DISTANZA 20
390 DATA TRIANGOLO
400 REM------------------ ROTAZIONE QUADRATO
405 DATA ALZARE
406 DATA AVANZARE 40
407 DATA ABBASSARE
408 DATA DISTANZA 20
410 DATA RIPETERE 8
420 DATA  QUADRATO
430 DATA  ROTAZIONE 45
440 DATA ANCORA
450 REM--------------- OTTAGONO
460 DATA ALZARE
462 DATA AVANZARE 40
464 DATA ABBASSARE
480 DATA RIPETERE 8
490 DATA  AVANZARE 10
500 DATA  ROTAZIONE 45
510 DATA ANCORA
850 DATA *
857 :
860 REM-----------LETTURA DEI COMANDI-
870 I=1
880 :
890 READ X$:PRINT X$
900 :
910 IF LEFT$(X$,4)="DEFI" THEN GOSUB 1720:GOTO 890
920 :
```

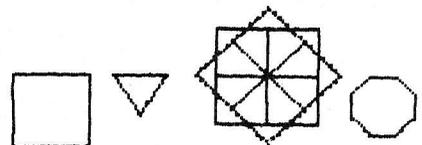
(segue)

(*seguito*)

```
930 IF X$="*" THEN NC=I:GOTO 1010
940 FOR K=1 TO 5
950 :IF X$=F$(K,0) THEN GOSUB 1940
960 NEXTK
961 :
962 :
963 :
964 :
965 :
970 :
980 CD$(I)=X$
990 I=I+1:GOTO 890
1000 :
1010 GOSUB 3050
1015 PL=1                    :REM OPERATORE RIGA IN CORSO
1020 :
1030 LG$=CD$(PL)
1040 C$=LEFT$(LG$,4)
1050 IF C$="RIPE" THEN GOTO 1470
1060 IF C$="ANCO" THEN GOTO 1510
1070 IF C$="AVAN" THEN GOSUB 1630
1080 IF C$="ROTA" THEN GOSUB 1540
1090 IF C$="DIST" THEN GOSUB 1390:IF V<>0 THEN DIST=V
1100 IF C$="DIS+" THEN GOSUB 1390:IF V<>0 THEN DIST=DIST+V
1120 IF C$="ALZA" THEN LV=1
1130 IF C$="ABBA" THEN LV=0
1175 PL=PL+1:IF PL>NC THEN 1185
1180 GOTO 1030
1182 :
1185 GET X$:GOTO 3430
1190 :
1380 REM--------------RICERCA SPAZIO--
1390 V=0
1400 FOR P=1 TO LEN(LG$)
1410 :IF MID$(LG$,P,1)=" " THEN V=VAL(RIGHT$(LG$,LEN(LG$)-P)):RETURN
1420 NEXT P
1430 P=0:RETURN
1440 REM---------------------- RIPETERE
1450 REM   PR:OPERATORE CATASTA RP() CONTENENTE
1455 REM      NO RIGA DOPO RIPETERE
1460 REM   NB:CONTIENE NO DI LOOPS
1470 GOSUB 1390
1480 PR=PR+1:RP(PR)=PL+1:NB(PR)=V
1490 PL=PL+1:GOTO 1030
1500 REM---------------------  ANCORA
1510 NB(PR)=NB(PR)-1:IF NB(PR)>0 THEN PL=RP(PR):GOTO 1030
1520 PR=PR-1:PL=PL+1:GOTO 1030
1530 REM---------------------- ROTAZIONE ANGOLO
1540 GOSUB 1390
1550 ANG=ANG+V
1560 IF ANG=>360 THEN ANG=ANG-360
1570 AR=ANG/360*6.28318
1580 CX=COS(AR):CY=SIN(AR)
1590 IF CX>-.01 AND CX<.01 THEN CX=0
1600 RETURN
1610 REM--------------DISEGNA LA RETTA
1630 GOSUB 1390
1640 IF P=0 THEN 1660
1650 DIST=V
1660 DX=DIST*CX:DY=DIST*CY
```

(*segue*)

(seguito)

```
1670 IF LV=1 THEN 1690
1680 :
1685 GOSUB 3340
1687 :
1690 X0=X0+DX:Y0=Y0+DY
1700 RETURN
1701 :
1702 :
1703 :
1704 :
1705 :
1710 REM---------------------CARICAMENTO FUNZIONI IN F$(,)
1720 NF=NF+1
1730 F$(NF,0)=RIGHT$(X$,LEN(X$)-5)
1740 FOR P=1 TO 10
1750 :READ F$(NF,P)
1760 :IF F$(NF,P)="FINE" THEN RETURN
1770 NEXT P
1780 STOP
1930 REM---------------------IMMISSIONE FUNZIONI
1940 P=1
1950 :
1960 IF F$(K,P)="FINE" THEN RETURN
1970 CD$(I)=F$(K,P)
1980 P=P+1:I=I+1:GOTO 1960
3000 ========================================
3040 :
3050 FOR I=832 TO 855:READ A:POKE I,A:NEXT I:SYS832
3060 :
3070 DATA 169,32,133,98,160,0,132,97,169
3080 DATA 0,145,97,200,208,251,230,98,165,98,201,64,208,241,96
3110 :
3120 PRINT CHR$(147)
3130 AGRA=8192:POKE 53272,PEEK(53272) OR 8
3140 :
3150 POKE 53265,PEEK(53265) OR 32    :REM COMMUTAZIONE GRAFICA
3180 :
3190 FOR I=1024 TO 1024+999:POKE I,3:NEXT I
3200 :
3210 FOR I=0 TO 7:P(I)=2↑I:NEXT I
3215 RETURN
3280 REM----------------- DISEGNO DI UN SEGMENTO
3340 FOR DD=0 TO DIST
3350 X=X0+DD*CX:Y=Y0+DD*CY
3360 C=INT(X/8):L=INT(Y/8):LG=Y AND 7
3370 OCT=AGRA+L*320+8*C+LG
3380 BIT=7-(X AND 7)
3390 POKE OCT,PEEK(OCT) OR P(BIT)
3400 NEXT DD
3410 RETURN
3420 REM----------------- RIPRISTINO
3430 GET X$:IF X$="" THEN 3430
3440 POKE 53272,PEEK(53272) AND 247
3450 POKE 53265,PEEK(53265) AND 223
```

I valori relativi dei bytes

• Permettono di disegnare figure di 24 × 21 punti gestiti dal Commodore 64; queste figure sono dislocate e visualizzate molto rapidamente.

- Possono essere gestiti simultaneamente 8 valori (da 0 a 7).
- L'impostazione dei valori relativi si compie linea per linea (21 linee di 24 punti). Ogni linea è codificata da tre valori decimali.
- La memoria V+21 permette di "ATTIVARE" uno stato ponendo 1 nel bit relativo (stato 2 per bit 2 per esempio).
- Le memorie V+23 e V+29 consentono di espandere un valore.

Esempio: 225 POKE V + 23,4: POKE V + 29,4
raddoppia le dimensioni del bit N. 2.

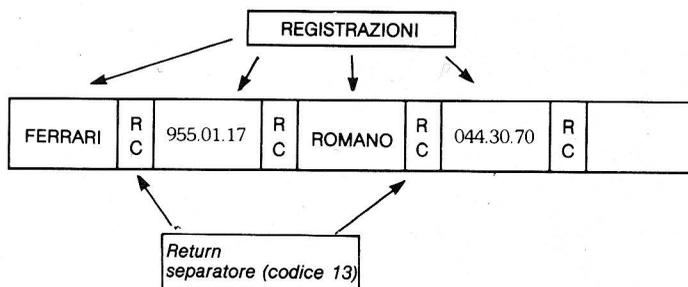
- I valori del disegno dell'esempio (farfalla) sono stati ottenuti grazie ad un "GENERATORE DI VALORI RELATIVI".

```
15   252 0
7    248 128
3    241 0
15   254 0
15   254 0
3    241 0
7    248 128
15   252 0
31   254 0
0    0   0
0    0   0
0    0   0
0    0   0
0    0   0
0    0   0
0    0   0
0    0   0
0    0   0

QWAS:DIREZIONI SPOSTAMENTO
S:SOLLEVARSI/ B:ABBASSARSI/ F:FINE
```

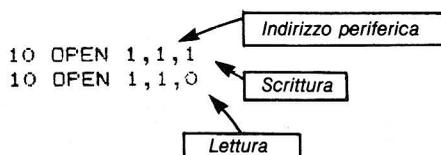
Farfalla

```
100 REM SPRITE
110 :
120 REM  DISEGNA UNA FARFALLA
130 :
140 PRINT CHR$(147)    :REM CANCELLAZIONE SCHERMO
150 V=53248            :REM INDIRIZZO      SPRITE
160 POKE V+21,4        :REM  SPRITE 2      ATTIVO
170 POKE 2042,13       :REM BLOCCO N.13
180 :
190 FOR I=0 TO 62      :REM LETTURA DATA
200 :READ D:POKE 13*64+I,D
210 NEXT I
220 REM---------------SPOSTAMENTO SPRITE
230 POKE 53281,1
240 FOR X=1 TO 200 STEP 2
250 :POKE V+4,X          :REM X
260 :POKE V+5,100
270 NEXT X
280 :
290 C=RND(1)*10
300 POKE V+41,C    :REM COLORE VARIABILE
310 GOTO 240
320 REM---------- DISEGNO SPRITE
330 DATA 0,0,0
340 DATA 15,252,0
350 DATA 7,248,128
360 DATA 3,241,0
370 DATA 15,254,0
380 DATA 15,254,0
390 DATA 3,241,0
400 DATA 7,248,128
410 DATA 15,252,0
420 DATA 31,254,0
430 DATA 0,0,0
440 DATA 0,0,0
450 DATA 0,0,0
460 DATA 0,0,0
470 DATA 0,0,0
480 DATA 0,0,0
490 DATA 0,0,0
500 DATA 0,0,0
510 DATA 0,0,0
520 DATA 0,0,0
530 DATA 0,0,0
```

Gli archivi sequenziali su cassetta

I files permettono di archiviare informazioni che possono essere in seguito consultate e modificate.
I supporti di questi files sono in genere magnetici (nastro o disco). Si distinguono **l'organizzazione sequenziale** e quella diretta.
Si parla di organizzazione sequenziale quando le informazioni sono archiviate "**sequenzialmente**" nell'ordine in cui arrivano.

Con questo tipo di organizzazione, un'informazione particolare può essere ritrovata solo dopo avere letto tutte le precedenti.
I "records" sono separati da < RETURN > (CODICE 13).
Le istruzioni di dialogo con i files (PRINT e INPUT) sono le stesse di dialogo con lo schermo.

Apertura: OPEN n. file, 1, $\begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$

- Un file sequenziale può essere aperto in modo "scrittura" od in modo "lettura".
- Deve essere chiuso prima di essere aperto in un altro modo.

Scrittura: PRINT # n. file, variabile

Scrive la variabile specificata sulla cassetta.

```
170 OPEN 2,1,1,"ESS"
180 :
190 INPUT "NOME (O FINE) ";NM$
200 INPUT "TELEFONO ";TEL$
210 :
220 PRINT 2,NM$
230 PRINT 2,TEL$
240 IF NM$="FINE" THEN CLOSE2:GOTO 260
250 GOTO 190
```

Lettura: INPUT # n. file, variabile

Trasferisce il valore presente in cassetta nella variabile specificata.

```
290 :
300 OPEN 2,1,0,"ESS"
310 INPUT 2,NM$,TEL$
320 :
330 PRINT NM$,TEL$
340 IF NM$="FINE" THEN CLOSE2:GOTO 370
350 GOTO 310
```

Chiusura: CLOSE n. file

Trasferisce il contenuto della "memoria tampone" su cassetta, liberando la prima.

ATTENZIONE: Le stringhe di caratteri non devono oltrepassare gli 80 caratteri.

File sequenziale su cassetta

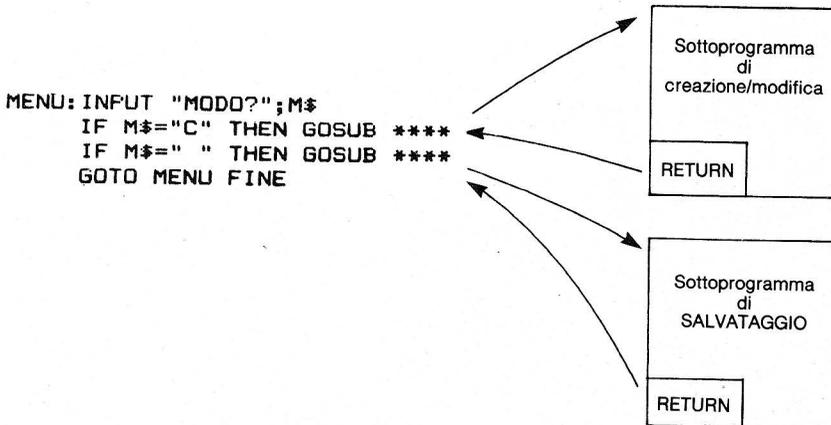
```
100 REM ARSEC  ARCHIVIO SEQUENZIALE SU  CASSETTA
110 :
130 :
150 PRINT
160 PRINT "SCRITTURA ARCHIVIO:PRINT
170 OPEN 2,1,1,"ESS"
180 :
190 INPUT "NOME (O FINE) ";NOM$
200 INPUT "TELEFONO ";TEL$
210 :
220 PRINT#2,NOM$
230 PRINT#2,TEL$
240 IF NOM$="FINE" THEN CLOSE2:GOTO 260
250 GOTO 190
255 :
260 INPUT "RIAVVOLGERE IL NASTRO E PREMERE RETURN";R$
270 REM------------------- LETTURA
280 PRINT "LETTURA ARCHIVIO":PRINT
290 :
300 OPEN 2,1,0,"ESS"
310 INPUT#2,NOM$,TEL$
320 :
330 PRINT NOM$,TEL$
340 IF NOM$="FINE" THEN CLOSE2:GOTO 370
350 GOTO 310
360 REM-------------- LETTURA CON GET#
370 INPUT "RIAVVOLGERE IL NASTRO E PREMERE RETURN> ";R$
375 OPEN 2,1,0,"ESS"
380 :
390 IF ST=64 THEN CLOSE2:END
400 GET#2,C$
410 :
420 PRINT C$,ASC(C$)
430 :
440 GOTO 390
```

Scrittura file

```
SCRITTURA FILE

PRESS RECORD & PLAY ON TAPE
OK
NOMÈ (O FINE) ? FERRARI
TELEFONO ? 112233
NOME (O FINE) ? ROSSI
TELEFONO ? 445566
NOME (O FINE) ? FINE
TELEFONO ? ***
RIAVVOLGETE E BATTETE <RETURN> ?
LETTURA FILE


PRESS PLAY ON TAPE
OK
FERRARI  112233
ROSSI    445566
FINE     ***
RIAVVOLGETE E BATTETE <RETURN> ?
```

Archivio di indirizzi (su file sequenziale)

Un archivio di indirizzi è gestito da tabelle in memoria centrale. Esse vengono poi memorizzate al termine delle operazioni in un file sequenziale letto in memoria centrale all'inizio della seguente sessione:

Naturalmente, le dimensioni dell'archivio sono relative allo spazio disponibile in memoria centrale. 20 kbytes permettono di gestire fino a 200 persone, se sono previsti in 100 caratteri per ciascuna di esse.
In caso di incidente (caduta di tensione, ad esempio), le modifiche non ancora trasferite sul file vanno perdute.
Organizzazione del programma: esso è scomposto in sottoprogrammi scelti per mezzo di un MENU.

Archivio di indirizzi

```
100 REM ARCHIND    ARCHIVIO INDIRIZZI
110 :
120 :
130 :
140 :
150 DIM NOM$(100),VIA$(100),CITTA$(100),CAP$(100)
160 DIM CLE$(100),IX(100)
170 :
180 PRINT CHR$(147):INPUT "NUOVO ARCHIVIO  (S/N) ";R$:IF R$="S" THEN 210
190 GOSUB 770
200 REM---
210 PRINT CHR$(147):PRINT "OPZIONE:":PRINT
220 PRINT TAB(3);"C   :CREAZIONE/MODIFICHE"
230 PRINT TAB(3);"LA  :LISTA ARCHIVIO"
240 PRINT TAB(3);"FIN :FINE SESSIONE (OBBLIGATORIA)"
250 PRINT:M$="":INPUT "OPZIONE";M$
260 IF M$="C" THEN GOSUB 310
270 IF M$="FIN" THEN GOSUB 650:END
280 IF M$="LA" THEN GOSUB 930
290 GOTO 210
300 REM=========CREAZIONE/MODIFICHE====
310 PRINT:NOM$="":INPUT "NOME ";NOM$:IF NOM$="" THEN RETURN
320 :
330 L=LEN(NOM$)
340 FOR RANG=1 TO 100
350 :IF NOM$(RANG)="" THEN 410
360 :IF NOM$=LEFT$(NOM$(RANG),L) THEN 470
370 NEXT RANG
380 STOP
390 :
400 REM------ NUOVO NOME
410 INPUT "NUOVO NOME  OK (S/N) ";R$
420 IF R$<>"S" THEN 310
430 :
440 NOM$(RANG)=NOM$
450 :
460 PRINT
470 PRINT VIA$(RANG);TAB(15);
480 VIA$="":INPUT "VIA ";VIA$:IF VIA$<>"" THEN VIA$(RANG)=VIA$
490 :
500 PRINT CITTA$(RANG);TAB(15);
510 CITTA$="":INPUT "CITTA' ";CITTA$:IF CITTA$<>"" THEN CITTA$(RANG)=CITTA$
520 PRINT CAP$(RANG);TAB(15);
530 CAP$="":INPUT "CAP ";CAP$:IF CAP$<>"" THEN CAP$(RANG)=CAP$
540 GOTO 310
550 :
560 :
570 :
580 :
590 :
600 :
610 :
620 :
630 :
```

(segue)

(seguito)

```
640 REM=================== SALVATAGGIO SU CASSETTA====
650 OPEN 2,1,1,"IND"
660 :
670 FOR I=1 TO 100
680 :IF NOM$(I)="" THEN PRINT#2,"FINE":CLOSE2:RETURN
690 :PRINT#2,NOM$(I)
700 :PRINT#2,VIA$(I)
710 :PRINT#2,CITTA$(I)
720 :PRINT#2,CAP$(I)
730 NEXT I
740 STOP
750 :
760 REM=================== LETTURA ARCHIVIO SU CASSETTA
770 OPEN 2,1,0,"IND"
780 :
810 :
830 :
840 FOR I=1 TO 100
850 :INPUT#2,X$:IF X$="FINE" THEN 900
860 :NOM$(I)=X$
870 :INPUT#2 ,VIA$(I),CITTA$(I),CAP$(I)
880 PRINT NOM$(I)
890 NEXT I
900 CLOSE2
910 RETURN
920 REM========= LISTA DELL'ARCHIVIO===
930 PRINT CHR$(147)
940 PRINT "LISTA DELL'ARCHIVIO ":PRINT
950 :
960 FOR F=1 TO 100
970 :IF NOM$(F)="" THEN 1030
980 :PRINT NOM$(F);TAB(13);
990 :PRINT CITTA$(F);TAB(30);
1000 :PRINT CAP$(F);
1010 :PRINT
1020 NEXT F
1030 PRINT:INPUT "PREMERE <RETURN> ";R$
1040 RETURN
```

Appendice 1

Ricapitolazione delle istruzioni BASIC

ABS (— 5)	Dà il valore assoluto di un numero PRINT ABS (— 5) → 5
AND	Permette la verifica di più condizioni: Le istruzioni dopo THEN sono eseguite SE la condizione 1 E la condizione 2 sono VERE 90 INPUT "X?": X 100 IF (X>0) AND (X<10) THEN PRINT "X è superiore a 0 ed inferiore a 10" Mette a zero un bit in un byte: POKE 53272, PEEK (52272) AND 247 Mette a zero il bit 3 del byte 53272
ASC ("B") ASC ("CARLO")	Dà il codice ASCII di un carattere o del primo carattere di una stringa (che non deve essere vuota) PRINT ASC ("B") → 66 PRINT ASC ("CARLO") → 67
ATN (X)	Fornisce l'arco tangente di X espresso in radianti
CHR$ (66)	Dà un carattere il cui codice è 66 Cioè B PRINT CHR$(66) visualizza il carattere B PRINT CHR$(14) passa in modo minuscolo/maiuscolo
CLOSE 9	Chiude il file logico 9
CMD 9	Trasferisce le uscite sul file logico OPEN 9,4: CMD 9: LIST scrive la lista su stampante
CONT	Riattiva l'esecuzione di un programma interrotta dal tasto "RUN/STOP", o STOP all'interno di un programma
COS	Fornisce il coseno di X espresso in radianti
DATA "gennaio", 83	Definisce dei dati che sono letti da READ 10 DATA "gennaio", 83 20 READ MESE$: READ AN 30 PRINT MESE$: AN → gennaio 83
DEF FN × (X) = ...	Definisce una funzione di X 10 DEF FNA(X) = X/2 20 PRINT FNA(12) → 6

(segue)

(seguito)

DIM MESE$(12) DIM MESE(12) DIM VEN(12,5)	Riserva lo spazio per una tabella MESE$( )
END	Definisce la fine del programma
EXP (X)	Dà l'esponente di X
FN (X)	Richiama la funzione definita da DEF
FOR I = 1 TO 5 NEXT I	Le istruzioni fra FOR e NEXT sono eseguite 5 volte 10 FOR I = 1 TO 5 20 PRINT I*I 30 NEXT I 1 1 2 4 . . 5 25
FOR I = 3 TO 1 STEP — 1 NEXT I	10 FOR I = 3 TO 1 STEP — 1 20 PRINT I*I 30 NEXT I 3 9 2 4 1 1
FRE (0)	Dà lo spazio libero in memoria centrale PRINT FRE(0) → 15234
GET X$	Legge un carattere da tastiera (senza riemetterlo)
GET# 9,A$	Riceve un carattere da una periferica aperta sul canale 9
GOSUB 1000	Provoca un allacciamento del programma alla riga 1000 (come GOTO 1000) ma quando si incontra una istruzione RETURN l'esecuzione prosegue alla 110 100 GOSUB 1000 1000 sotto-programma 110 1010 200 GOSUB 1000 1020 210 1030 RETURN
GOTO 1000	Provoca un allacciamento del programma alla 1000 Può essere utilizzato in modo diretto dopo una interruzione, le variabili non sono riportate a zero (come farebbe RUN 1000)
IF cond. THEN istr.	Testa una condizione ed esegue la o le istruzioni dopo THEN se la condizione è VERA 10 INPUT "Età?"; ETA 20 IF (ETA>18) THEN PRINT "Siete maggiorenni"
INPUT "ANNO?"; AN INPUT "NOME?"; N$ INPUT "NOME, ETÀ"; N$, ETA INPUT# 9, A$	Emette il messaggio "ANNO?" ed attende che l'operatore immetta il valore di AN Emette il messaggio "NOME, ETÀ" ed attende che l'operatore immetta i valori di NOME e di ETA (separati da un virgola). Una stringa non deve contenere delle virgole. Riceve una stringa da una periferica aperta sul canale 9
INT (valore)	Dà la parte intera di un valore PRINT INT(12,6) → 12
LEFT$ ("BASIC", 3)	Dà i caratteri di sinistra di una stringa PRINT LEFT$ ("BASIC", 3) → BAS

(segue)

(seguito)

LEN ("BASIC")	Dà la lunghezza di una stringa di caratteri PRINT LEN ("BASIC") → 5
LIST LIST 10 LIST 10—30 LIST —30 LIST 30—	Lista tutto il programma Lista la riga 10 Lista le righe da 10 a 30 Lista le righe da 0 a 30 Lista le righe da 30 alla fine
LOAD "INIZIO"	Carica in memoria centrale il programma "INIZIO"
LOG (X)	Dà il logaritmo in base 10
MID$ (X$, 7, 4)	Dà i caratteri centrali di una stringa 10 X$ = "BASIC PER TUTTI" 20 PRINT MID$ (X$, 7, 3) → PER
NEW	Cancella il programma presente in memoria centrale
NEXT	Fine del loop (vedere FOR)
OPEN 9, 4	Apre un file logico per una periferica Apre il canale 9 per la stampante
ON B GOSUB 100, 200, 500	Il programma esegue la 100, la 200 o la 500, a seconda del valore di B (1, 2, 3). Al ritorno dal sottoprogramma, il programma continua alla riga che segue ON B GOSUB... 10 INPUT "Scelta? (1, 2, 3)": SC 20 ON SC GOSUB 100, 200, 500 30 END 100 PRINT "Scelta 1": RETURN 110 : 200 PRINT "Scelta 2": RETURN 210 : 500 PRINT "Scelta 3": RETURN
ON B GOTO 100, 200, 500	La stessa cosa di ON B GOSUB... ma senza ritorno
OR POKE 53272, PEEK (53272) OR8	Permette il test di più condizioni. Le istruzioni dopo THEN sono eseguite se la cond. 1 O la cond. 2 è VERA. 90 INPUT "X?"; X 100 IF (X = 1) OR (X = 3) THEN PRINT "X è uguale a 1 o 3" Mette a 1 il bit 3 del byte 53272
PEEK (1000)	Fornisce il contenuto del byte di memoria 1000 10 X = PEEK (1000)
? PEEK (214)	Stampa il contenuto del byte di memoria 214
POKE 214,10	Colloca 10 nel byte di memoria 214
POS. (X)	Fornisce la posizione del cursore nella linea
PRINT PRINT "BUONGIORNO" PRINT "ROSSI"; TAB (9); "A."	Provoca un salto di linea Visualizza BUONGIORNO Visualizza ROSSI e poi A. alla colonna 9
PRINT# 9, A$	Trasmette alla periferica aperta il canale 9
READ MESE$, ANNO	Vedi DATA

(segue)

(seguito)

REM blabla	Tutto ciò che segue REM è commento
RESTORE	Posiziona il puntatore delle DATA all'inizio Permette così la rilettura
RETURN	Segnala la fine di un sottoprogramma. Provoca un ritorno del programma alla riga dopo GOSUB
RIGHT$ (X$, 3)	Dà i caratteri di destra di una stringa 10 X$ = "BASIC" 20 PRINT RIGHT$ (X$, 3) → SIC
RND (Y)	Fornisce un numero a caso compreso tra 0 e Y
RUN RUN 100	Provoca l'esecuzione di un programma Imposta i valori delle variabili a zero Provoca l'esecuzione del programma dalla riga 100
SAVE "PROG"	Salva il programma PROG sulla cassetta
SGN (X)	Dà il segno di numero: 1 positivo 0 nullo — 1 negativo
SIN (X)	Dà il seno di X espresso in radianti
SPC (3)	Stampa 3 spazi: 10 PRINT "XXX", SPC (3); "XXX" → XXX XXX
SQR (X)	Dà la radice di X che deve essere positivo
STOP	Blocca l'esecuzione che può essere proseguita battendo CONT
STR$ (123)	Converte un numero in stringa: 10 X = 123: X$ = STR$ (X) 20 PRINT RIGHT$ (X$, 1) → 3
SYS SYS 832	Lancia un programma in linguaggio macchina Lancia un programma in linguaggio macchina che inizia all'indirizzo 832
TAB (X)	Stampa con uno spazio di X caratteri all'inizio di riga
TAN (X)	Dà la tangente di X espressa in radianti
THEN	(Allora) Vedi IF
TO	Condiziona la fine di un loop (vedi FOR)
VAL (X$)	Dà il valore numerico di una stringa: 10 X$: "123 LIRE" 20 PRINT VAL (X$) → 123
WAIT 198,1	Attende che si batta un tasto
VERIFY	Confronta il programma in memoria con la sua copia sulla cassetta

Appendice 2

Codici ASCII

- **Caratteri ottenuti per mezzo di PRINT CHR$ (codice)**

0		32		64	@	96	─
1		33	!	65	A	97	♠
2		34	"	66	B	98	│
3		35	#	67	C	99	─
4		36	$	68	D	100	─
5	BIANCO	37	%	69	E	101	─
6		38	&	70	F	102	─
7		39	'	71	G	103	│
8		40	(	72	H	104	│
9		41	)	73	I	105	╮
10		42	*	74	J	106	╰
11		43	+	75	K	107	╯
12		44	,	76	L	108	└
13	RETURN	45	−	77	M	109	╲
14		46	.	78	N	110	╱
15		47	/	79	O	111	┌
16		48	0	80	P	112	┐
17	CRSR ↓	49	1	81	Q	113	●
18	RVS ON	50	2	82	R	114	─
19	CLR/HOME	51	3	83	S	115	♥
20		52	4	84	T	116	│
21		53	5	85	U	117	╭
22		54	6	86	V	118	╳
23		55	7	87	W	119	○
24		56	8	88	X	120	♣
25		57	9	89	Y	121	│
26		58	:	90	Z	122	♦
27		59	;	91	[	123	┼
28	ROSSO	60	<	92	£	124	▒
29	CRSR →	61	=	93	]	125	│
30	VERDE	62	>	94	↑	126	π
31		63	?	95	←	127	◥

(segue)

(seguito)

128		160	
129		161	
130		162	
131		163	
132		164	
133	F1	165	
134	F2	166	
135	F3	167	
136	F4	168	
137	F5	169	
138	F6	170	
139	F7	171	
140	F8	172	
141	SHIFT RETURN	173	
142	MAIUSCOLO	174	
143		175	
144	NERO	176	
145	CRSR ↑	177	
146		178	
147	CLR/HOME SHIFT	179	
148	INST DEL	180	
149	MARRONE	181	
150	ROSSO CHIARO	182	
151	GRIGIO 1	183	
152	GRIGIO 2	184	
153	VERDE CHIARO	185	
154	BLU CHIARO	186	
155	GRIGIO 3	187	
156	PORPORA	188	
157	← CRSR	189	
158	GIALLO	190	
159	TURCHESE	191	

Appendice 3

Codici video

• I seguenti codici video sono ottenuti per mezzo di:

POKE 1024 + Y*40+X, codice
POKE 55296 + Y*40+X, colore

POKE 53272,21 permette di ottenere la combinazione 1 (maiuscolo/grafico)
POKE 53272,23 permette di ottenere la combinazione 2 (maiuscolo/minuscolo)

POKE	MODO1	MODO2	POKE	MODO1	MODO2	POKE	MODO1	MODO2
0	@		32			64	─	
1	A	a	33	!		65	♠	A
2	B	b	34	"		66	│	B
3	C	c	35	#		67	─	C
4	D	d	36	$		68	─	D
5	E	e	37	%		69	─	E
6	F	f	38	&		70	─	F
7	G	g	39	'		71	│	G
8	H	h	40	(		72	│	H
9	I	i	41	)		73	╮	I
10	J	j	42	*		74	╰	J
11	K	k	43	+		75	╯	K
12	L	l	44	,		76	L	L
13	M	m	45	-		77	╲	M
14	N	n	46	.		78	╱	N
15	O	o	47	/		79	┌	O
16	P	p	48	0		80	┐	P
17	Q	q	49	1		81	●	Q
18	R	r	50	2		82	─	R
19	S	s	51	3		83	♥	S
20	T	t	52	4		84	│	T
21	U	u	53	5		85	╭	U
22	V	v	54	6		86	╳	V
23	W	w	55	7		87	○	W
24	X	x	56	8		88	♣	X
25	Y	y	57	9		89	│	Y
26	Z	z	58	:		90	♦	Z
27	[		59	;		91	┼	
28	£		60	<		92	▒	
29	]		61	=		93	│	
30	↑		62	>		94	π	
31	←		63	?		95	◥	

(segue)

Appendice 4

Messaggi di errore

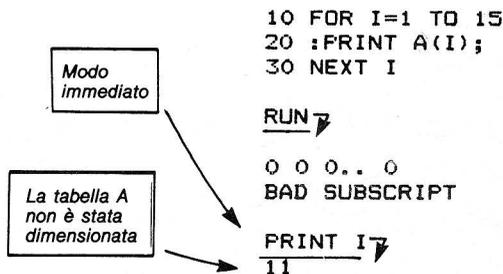
Se il messaggio inviato dal Basic non è sufficiente a trovare l'errore, provate a **visualizzare in modo diretto** i valori delle variabili (prima di modificare il programma, poiché una modifica imposterebbe le variabili a zero).

BAD SUBSCRIPT

Si cerca di fare riferimento ad un elemento al di fuori di una tabella.

```
10 DIM A(13)
   :
70 PRINT A(15)
```

Occorre dimensionare la tabella con 15 elementi.

Lo stesso errore si ha quando la tabella non è stata dimensionata con DIM (e viene quindi dimensionata a 10 dal BASIC). In questo caso, PRINT A (11) provoca un errore.

CAN'T CONTINUE (non può continuare)

L'esecuzione del programma non può essere proseguita:

— Una istruzione è stata modificata.

DIVISION BY ZERO (divisione per zero)

Divisione per zero impossibile (aggiungere un test IF X = 0 THEN...)

EXTRA IGNORED

Si cerca di introdurre un dato non previsto.

È accettato solamente 11 (non immettere la virgola).

ILLEGAL DIRECT

Questa istruzione non può essere utilizzata in modo diretto (p. es. INPUT).

NEXT WITHOUT FOR (NEXT senza FOR)

Un'istruzione NEXT incontrata senza che sia stata prima eseguita una FOR.

Occorre fare: 100 GOTO 190

OUT OF DATA (DATA in eccesso)

Si cerca di leggere delle DATA con READ quando sono state già lette tutte.

OUT OF MEMORY (spazio in memoria completo)

OVERFLOW (straripamento)

Il valore di un numero oltrepassa la capacità prevista.

REDIMENSIONED ARRAY (ridimensionamento di tabella errato)

Si cerca di dimensionare una tabella già dimensionata.
È già stato fatto riferimento ad un elemento della tabella (il che provoca il suo dimensionamento a 10 da parte del Basic), e poi la si dimensiona esplicitamente.

```
10 A(5)=123
20 DIM A(15)

RUN

REDIMENSIONNED ARRAY
```

REDO FROM START (ricominciare dall'inizio)

Si cerca di inserire una stringa in un' istruzione INPUT quando si attende un valore numerico.

RETURN WITHOUT GOSUB (ritorno senza richiamo)

Si trova un'istruzione RETURN senza che sia stata eseguita una GOSUB in precedenza.

```
10 GOSUB 100
20 :
100 PRINT "SOTTO PROGRAMMA"
110 RETURN
```

È stato dimenticato END o STOP dopo la riga 10: il sottoprogramma viene eseguito 2 volte. (Bisogna aggiungere: 20 STOP).

STRING TOO LONG (stringa troppo lunga)

Una stringa diventa superiore a 255 caratteri.

```
10 X$=X$+"A"
20 GOTO 10
```

SYNTAX ERROR (errore di sintassi)

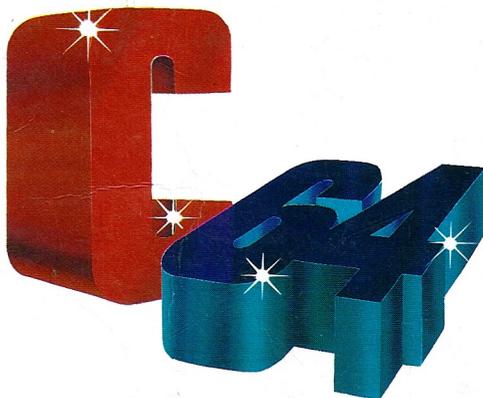
TYPE MISMATCH

Si cerca di attribuire un valore numerico ad una stringa.
10 X$ = 123 (si deve fare X$ = "123" o X = 123)

10 X = "ROSSI" (si deve fare X$ = "ROSSI")

UNDEF'D STATEMENT (n. riga indefinito)
Un'istruzione GOTO o GOSUB fa riferimento ad una riga inesistente.

UNDEF'D FUNCTION (funzione indefinita)
Si richiama una funzione che non è stata definita.

Finito di stampare
nel mese di novembre 1984
dal Centro Grafico Linate - San Donato Milanese

Imparate a usare il Commodore 64:
con "Il Commodore 64 per tutti" in mano, metterevi davanti alla vostra tastiera e incominciate a scrivere alcune istruzioni.

Rapidamente assimilerete le nozioni fondamentali della programmazione:
le variabili, i test, i loop ...
Grazie ai numerosi esempi illustrati e ai programmi commentati potrete accostarvi alla programmazione del Commodore 64, alla grafica, ai suoni, ai suggestivi sprite.

Su queste basi vi sarà facile approfondire le vostre conoscenze e scrivere i vostri programmi di gestione, didattici e di giochi.

★

I.S.B.N. 88.7688.500.5

IL COMMODORE 64 *per tutti* INIZIAZIONE + PROGRAMMI J. BOISGONTIER · S. BREBION · G. FOUCAULT

EPSI